(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
14 December 2000 (14.12.2000)

PCT

(10) International Publication Number
WO 00/75775 A2

(51) International Patent Classification[7]: G06F 9/445

(21) International Application Number: PCT/US00/15689

(22) International Filing Date: 7 June 2000 (07.06.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/328,058 8 June 1999 (08.06.1999) US

(71) Applicant *(for all designated States except US)*: THINKPULSE, INC. [US/US]; 3 Lagoon Drive, Suite 300, Redwood City, CA 94065-1566 (US).

(72) Inventor; and

(75) Inventor/Applicant *(for US only)*: FARRUGIA, Augustin [FR/US]; 1232 Lisa Lane, Los Altos, CA 94024 (US).

(74) Agent: ADELI, Mani; Stattler Johansen & Adeli LLP, P.O. Box 51860, Palo Alto, CA 94303-0728 (US).

(81) Designated States *(national)*: AE, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CR, CU, CZ, DE, DK, DM, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

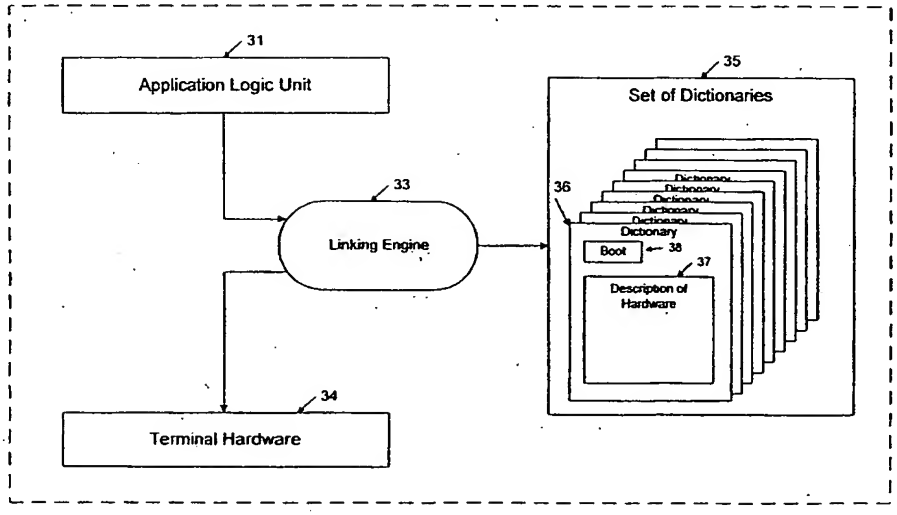
(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:

— *Without international search report and to be republished upon receipt of that report.*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: METHOD AND SYSTEM OF LINKING A SMART DEVICE DESCRIPTION FILE WITH THE LOGIC OF AN APPLICATION PROGRAM



(57) Abstract: A linking engine providing a process of matching the logic of an application, described in an application logic unit, whith the hardware specific description of the application for use with applications run on smart card terminals. The hardware specific description of the application is contained in dictionaries. The linking engine tests dictionaries for compatibility with the application logic unit by runnning a process that returns a predetermined response only when the dictionary describes the hardware implementation used in the test. A dictionary outputting the correct response is linked by the linking engine to provide the description of the hardware implemented in running the logic of the application.

WO 00/75775 A2

METHOD AND SYSTEM OF LINKING A SMART DEVICE DESCRIPTION FILE WITH THE LOGIC OF AN APPLICATION PROGRAM

BACKGROUND

Field of the Invention

The present invention relates to the field of smart devices. More particularly, the present invention relates to the field of application software used in connection with a smart device.

Related Background

Smart devices, such as smart cards, smart watches, smart tags, or other portable objects with the ability to either process data or store data, interface with other processing devices to provide a functional implementation. Examples of functional implementations, or uses, of the smart device include stored value (often referred to as an electronic purse), loyalty award programs, secure access, secure authentication, as well as many other uses.

Figure 1 is a block diagram of a conventional smart device system (1). A smart device (2) is in communication with a terminal (3). Application software that runs on the terminal (3) is often referred to as an "application" (4). Applications (4) interfaces with the terminal hardware (5) through an Application Programming Interface (API) (6). The API is assembled of smart device interfaces that support applications. Typically, the API runs on top of a layer of software (not shown) which interacts directly with the terminal hardware. Examples of such layers include the Open Card Framework (OCF) layer, the Personal Computer to Smart Card (PC-SC) layer, and the Sun™ initiative layer.

Application software, often referred to as "applets" (7), for processing data on the smart device (2) run on the smart device and interact with the smart device hardware (8) through the smart device operating system (OS) (9). The application (4) running on the terminal (3) interfaces with the applet (7) through the API (6), terminal hardware (5), and the smart device hardware (8).

Data is downloaded/uploaded to the smart device (2) by the terminal (3). The terminal can be either of the contact or contactless type. In contact type smart devices contact tabs of the terminal establish communication with the smart device through physical contact with contact pads located on the smart device. In contactless terminals radio frequency (RF) is typically used to provide communication with the smart device. Other contactless terminals can use optical, microwave, or other communication methods.

Because memory space is at a premium in terminals, conventional APIs are optimized to support a specific functional implementation of a particular terminal-device combination. For example, an API designed for use with a smart device functioning as an electronic purse is optimized to support those functions commonly used in implementing electronic purse applications. Similarly, an API designed to support loyalty award applications, such as airline frequent flyer rewards program, is optimized to support those functions commonly used in implementing loyalty applications. Electronic purse applications would typically not be interoperable with loyalty APIs as such APIs would not support the electronic purse functionality. Similarly, loyalty applications and applets are typically not interoperable with electronic purse APIs. In this manner applications intended for a given functional implementation, such as loyalty, are typically only interoperable with APIs for the same functional implementation.

APIs are typically optimized to work with a particular smart device from a particular manufacturer. An API for a particular manufacturer's electronic purse smart device supports different application functionalities than an electronic purse API designed for another manufacturer's smart device. Additionally, even different models of electronic purse smart devices from a particular manufacturer may require a different APIs that are not interoperable with applications written for the other models of electronic purse smart devices from the same manufacturer. APIs are also typically optimized to work with a particular reader. Thus, an application written for a given API may be capable of being used with a small subset of available smart devices in combination with a particular set of readers. This results in a lack

of interoperability of applications written for loyalty across APIs for cards from different manufacturers and for particular terminals.

This lack of interoperability of applications across the proliferation of APIs optimized for specific smart devices, terminals and functionalities reduces the ability for developers to create applications that can be implemented on a wide variety of smart devices. The present design of conventional APIs requires developers to write and develop applications for use with a specific, or a specific set, of APIs. Consequently, applications written in this manner are only useful with a limited number of smart devices and terminals.

Existing programming interfaces have attempted to solve the problems associated with the lack of flexibility with conventional APIs. One attempt at providing greater interoperability among applications and APIs is the Java programming language.

Figure 2 is a block diagram illustrating the architecture of a typical Java™ card system (10). A Java card (11) is a conventional smart device that may run Java applets (12). Java applets (12) are executed by the smart card Java card virtual machine (JCVM) (13). The JCVM (13) runs on top of the smart card's OS (14). The OS (14) interfaces with the hardware (15) of the smart card. Java applications (16) run on the JVM (17) of the terminal (18). The terminal (18) is in communication with the Java card (11) can be either a contact or contactless terminal. Applications interface with the hardware of the terminal (19), the hardware of the Java card, and the Java applet through the API (20). Like the API (6) of **Figure 1**, the API (20) interfaces with the application to support the functionality of the intended use of the Java card. Because of the premium on memory of the terminal, the API typically is optimized to support the functionality for the intended use of the smart card. Thus, an API optimized for electronic purse applets would typically not be able to support loyalty applications. Similarly, an API optimized for loyalty would not be able to support electronic purse applications. In this manner, the Java card has the same disadvantages of the conventional smart card architecture shown in **Figure 1**.

Attempts to allow interoperability of applications, in either the conventional architecture of **Figure 1**, or the Java card architecture of **Figure 2**, result in a loss of supporting functions for applications written for the intended use of the card. The architectures of the presently available smart card systems force designers to make a trade off between interoperability and functionality, with most systems choosing functionality over interoperability.

Attempts to increase the interoperability of conventional APIs by increasing the functions the API supports has the disadvantage of increasing the footprint of the API, i.e. the memory required to store the API on the terminal. This has the disadvantage of increasing the cost and complexity of the terminal.

The conventional architecture of applications and APIs also has the disadvantage that when a new smart device or terminal is introduced, often with a new functionality, existing applications often are not interoperable with an API intended for the new smart device or terminal. An API released prior to the development of the new functionality would not support new functionalities.

Brief Description of the Figures

Figure 1 is a block diagram depicting the application system architecture of a conventional smart card.

Figure 2 is a block diagram depicting the application system architecture of a conventional Java™ card system.

Figure 3 is a block diagram of an application for the smart device including an application logic unit and an application protocol, in accordance with the present invention.

Figure 4 depicts the architecture of the application and linking engine, in accordance with the present invention.

Figure 5 is a flow chart illustrating the process implemented by the linking engine in running the application, in accordance with the present invention.

Figure 6 is a flow chart illustrating the process of selecting the appropriate dictionary for the hardware implementation used at the run time of the application logic unit, in accordance with the present invention.

Figure 7 is flow chart illustrating the process of running an application in accordance with the present invention.

SUMMARY

The present invention provides a linking engine for smart devices which links the logic of the application, the application logic unit, to a hardware specific description of the application, the application protocol. An application logic unit is written in a conventional computer language and expresses the logic of the application without regard for the specific hardware implementation of the terminal and smart device. During run time the linking engine looks up the appropriate dictionaries corresponding to the hardware elements. Each dictionary has a boot process which, when run, outputs a predetermined response when the hardware used is the hardware the dictionary describes. In this manner dictionaries retrieved

from a set of dictionaries are tested to find the dictionary appropriate for the terminal and smart device present when the application logic unit is to be run.

DETAILED DESCRIPTION

The present invention is described in the context of a specific embodiment. This is done to facilitate the understanding of the features and principles of the present invention and the present invention is not limited to this embodiment. In particular, the present invention is described in the context of a smart device. Examples of smart devices applicable to the present invention include, without limitation, smart cards, smart watches, smart tags, smart wristbands, and smart pendants. Smart devices can be either of the processor type, where the device has the ability to run applets to process data, or of the memory type, where the device is sued to store data. Additionally, the present invention is described in the context of a terminal. A terminal can be any computing device capable, by itself or with other devices, of communicating with a smart device. Examples of terminals include, without limitation, personal computers, server computers, hand-held computers, point of sale terminals, portable phones and communication devices, and computer networks.

In the following figures like objects are provided with the same identifying number as an aid in understanding the present invention.

Figure 3 is a block diagram depicting the architecture of an application program (application) (30) in accordance with the present invention. The application typically runs on a terminal in communication with a smart device. Applications can be any implementation of the processing of data. Typical examples of applications include the debit functions in an electronic purse card, credit and add loyalty points in a frequent flyer program card, key generation in a security authorization card, memory retrieval in a medical history card, or security and memory functions common to many card implementations.

The application (30) of the present invention is segregated into two categories of software to form the application. The application (30) is composed of an application logic unit (31) and an application protocol (32). The application logic unit is the logic of the application, independent of the hardware implementation. The application protocol is a hardware specific component of the application and provides the data and rules necessary to implement the logic of the application protocol on a specific hardware implementation.

Figure 4 is a block diagram of the architecture of the application logic unit (31), linking engine (33) and terminal hardware (34). The application logic unit interacts with the linking engine to link an application protocol to the application logic unit. The linking engine interacts with a set of dictionaries (35) containing at least one dictionary (36). A dictionary (36) includes a description (37) relating to the terminal hardware (34) and to the smart card hardware (not shown). More particularly, the dictionary includes a series of verbs (not shown). The verbs are described in the body of the dictionary, the description being the definition of the verb. The definition of the verb is the hardware specific component of the application. The set of dictionaries (35) are specified in a boot file (38). The boot file is used by the linking engine (33) in testing the dictionary to determine whether that dictionary is the appropriate dictionary for the smart device and terminal present at the run time of the application. The appropriate dictionary is used as the application protocol which, when linked to the application logic unit, provides the hardware specific component of the application

Figure 5 is a flow chart of the process the linking engine utilizes in running an application. In the presently preferred embodiment of the present invention the linking engine has been established and the application logic unit has initiated a boot check request prior to step (40), as described below in connection with **Figure 7**. At step (40) the linking engine waits for an indication that a smart device is in communication with the terminal. When a positive indication is received, i.e. when there is a smart device in communication with the terminal, the linking engine proceeds to step (41) to initiate a compatibility test for

the dictionary to use as the application protocol. The process of selecting a dictionary for use as the application protocol is described below in connection with **Figure 6**. At step (42) the linking engine receives from the application protocol selection process the identifier of the compatible dictionary to use as the application protocol. The dictionary corresponding to the identifier is used as the application protocol. The linking engine then proceeds to step (43) to wait for a method call from the application logic unit.

In response to a method call from the application logic unit the linking engine proceeds to step (44) to look-up the method in the application protocol. At step (45) the linking engine determines whether a verb corresponding to the method exists in the application protocol. If the verb does not exist, the linking engine returns to step (44) to wait for a method call from the application logic unit. If the verb does exist in the application protocol, the linking engine proceeds to step (46) where the linking engine retrieves the definition from the application protocol and links the definition to the method call from the application logic unit. The linking engine then uses the verb's definition at step (47) in executing the method call from the application logic unit. The description in the application protocol of the hardware is used to generate responses to the method call. The responses generated at step (47) from executing the method call according to the rules and data of the definition are then returned to the application logic unit at step (48). The linking engine returns to step (43) to wait for a method call from the application logic unit.

The linking engine may return a critical fault to the application logic unit in response to any condition which jeopardizes the proper execution of the application logic unit. The conditions which may give rise to a critical fault include the absence from the application protocol of a critical definition, absence from the application protocol of critical rules or data, absence of critical data from the application logic unit being passed to the method call, problems with the data used in the method call, or an inappropriate responses to executing the method call. In the preferred embodiment of the present invention, the conditions that trigger a critical fault are included in the application protocol.

Figure 6 is a flow chart illustrating the process used by the linking engine in selecting a dictionary for use as the application protocol with a given terminal and smart device. At step (50) the linking engine receives a boot check request, or dictionary selection request, from an application logic unit. In response, at step (51) the linking engine retrieves a list of potentially compatible dictionaries. In the presently preferred embodiment of the present invention, the list of potentially compatible dictionaries includes those dictionaries specified as potentially compatible with the application logic unit, the terminal and the smart device. More preferably, the dictionaries included on list of potentially compatible dictionaries are pre-selected to include dictionaries that are compatible with the terminal and application logic unit. In such a case, the list of dictionaries is inclusive of all the dictionaries that may be needed to provide a description compatible with any smart devices available for use with the application and terminal.

The list of dictionaries is located on a file separate from the application logic unit. This file is referred to as the boot file. As the boot file is separate from the application logic unit this file can be updated to add dictionaries, as when a new smart device is introduced, without the need to modify the application logic unit. Additionally, the boot file includes an address specifying where to find the dictionary. Depending on the implementation of the terminal and smart device the dictionary can be located on a computer accessible by the terminal through a network or Internet connection, or the dictionary can be stored in the memory of the smart device. When the terminal is able to connect to a network, or to the Internet, storing the dictionary remote from the smart device reduces the memory requirements of the smart device. Additionally, remote storage of the dictionary allows additional dictionaries to be added to the boot file without having to download dictionaries to the smart device. Remote storage of the dictionary also allows the dictionary to be modified, should this deem desirable. Alternatively, storage of the dictionary on the smart device provides additional assurance that the dictionary has not been modified, especially when the dictionary is stored in write resistant memory. Storage of the dictionary on the smart device

allows access of the dictionary when the terminal is not able to access a network, or the Internet. In one embodiment of the present invention, a core set of dictionaries is stored on the card and an enhanced set of dictionaries is stored in a remote file. The core dictionaries support the basic functionality of a given card. The enhanced dictionaries provide support to additional features available for use with the card. In this manner the card provides basic support common with off-line applications, in addition to providing enhanced functionality common with on-line applications.

At step (52) the linking engine selects one of the dictionaries listed in the boot file for compatibility testing. At step (53) the linking engine retrieves the dictionary selected at step (52). At step (54) the linking runs the boot to scan the retrieved dictionary to retrieve selected segments of the dictionary for input to the linking engine for compatibility testing. The boot file provides a description of the segments of the dictionary to be scanned and used in generating the output value of the compatibility test. The linking engine generates an output value based on the specified dictionary information. At step (55) the linking engine receives the output value from the compatibility test. The output value is compared to a bootcheck value at step (56). In the preferred embodiment of the present invention the bootcheck value is a standard value common for all smart devices. The bootcheck value may be stored in the application logic unit, the boot file, or the dictionary. If the output value is the same as the bootcheck value the linking engine adds the dictionary corresponding to the output value to a list of compatible dictionaries at step (57). The linking engine then proceeds to step (58). If at step (56) the output value is not the same as the bootcheck value the linking engine proceeds to step (58) without adding the dictionary corresponding to the boot to the compatible dictionary list.

Different applications and smart devices may require testing of multiple functionalities of the dictionary and the card. In such instances, there may be a plurality of bootcheck values corresponding to the multiple functionalities to be tested for compatibility. In the preferred embodiment of the present invention, the bootcheck value corresponding to

each functionality tested will be common for all smart devices. Optionally, the linking may generate a plurality of output values based on the boot file, the output values being dependent on the smart device. At step (55) the linking engine receives the output values from the compatibility test. The output values are compared to the corresponding bootcheck values at step (56). If the output value is the same as the bootcheck value corresponding to the same functionality being tested the linking engine adds the dictionary corresponding to the output value to a list of compatible dictionaries at step (57). The linking engine then proceeds to step (58). If at step (56) the output value is not equal to the bootcheck value the linking engine proceeds to step (58).

At step (59) the lining engine checks whether all of the dictionaries listed in the boot file have been tested for compatibility. If all of the dictionaries have not been tested, the system returns to step (52). If all of the dictionaries have been tested, the linking engine proceeds to step (59). In the presently preferred embodiment of the present invention the linking engine tests all of the dictionaries and compares the output value form each dictionary against the bootcheck value. All the dictionaries having an output value equal to the bootcheck value are added to the list of compatible dictionaries.

Optionally, the linking engine can compare the dictionaries on the list of compatible dictionaries to select which dictionary to specify as the application protocol. Selection could be based on the smallest file size of the dictionary, the most features of the dictionary, or some other criteria. Alternatively, as all of the dictionaries are suitable for use as the application protocol, the linking engine may select any of the dictionaries from the list at random. This selection process, by any suitable process, is performed at step (59).

At step (60) the linking engine returns the identifier of the dictionary selected for use as the application protocol to the application logic unit.

Figure 7 is flow chart illustrating the process of running an application in accordance with the present invention. At step (70) the application logic unit begins the process of creating a virtual smart device by creating a profile for the dictionary. At step (71) the

application logic unit establishes the linking engine to run on top of any software layer which interacts with the terminal hardware. In the preferred embodiment of the present invention the linking engine is stored on the terminal. Step (71) makes an initial call to the linking engine. At step (72) the virtual smart device is created as running on top of the terminal hardware layer of software and referencing the profile for the dictionary. Creating a virtual smart device allows the application logic unit to access the linking engine to link the application logic unit to an application protocol.

At step (73) the application logic unit initiates a boot check to find the dictionary to be used as the application protocol. The boot check process is described in detail above in connection with **Figure 6**. The dictionary selected during the boot check process is returned at step (74) and is used as the application protocol.

At step (75) the application logic unit invokes a method call. A method call can be any logical process to access or manipulate data on either the smart device or the terminal. Checking the balance in the card's electronic purse is one example of a method call. Another example of a method call is the debiting of loyalty points in a smart card. As described in connection with **Figure 5**, the linking engine receives the invoked method call at step (43) and uses the dictionary definition of the method in running the application. Based on the definition contained in the dictionary, at step (47) the linking engine runs a sequence of smart device commands. These commands output responses based on the data from the card and the dictionary. The linking engine returns the responses related to the smart device commands to the application logic unit at step (48) of **Figure 5**. The application logic unit receives the command output responses at step (76). The application logic unit then continues by either invoking additional method calls, in which the process begins again at step (75), or by ending.

Example

A source code example of an application logic unit and the appropriate dictionary used as the application protocol, in accordance with the present invention, is given below. The boot file for testing compatibility between a dictionary and the application logic unit is also provided. The example application logic unit is written in Java™ and the related application protocol and boot files are written in XML. In the presently preferred embodiment, XML is chosen for its suitability in providing descriptions of the hardware implementation.

In the examples below the line numbers (A1, A2, A3, ...; B1,B2, B3, ...; C1, C2, C3,....) are only used for discussion purposes and are not part of the source code. These line numbers refer only to the line as printed herein and not to a "line" of the source code.

APPLICATION LOGIC UNIT

The Java example below of an application logic unit is for a loyalty application, as indicated by line A1. This example of an application logic unit checks the loyalty points stored on the smart card, determines whether additional loyalty points are to be credited and added to the loyalty points total, and whether loyalty points are being redeemed and debited from the loyalty points total. At line A31 the string variable for the dictionary is created. The main part of the program begins at line A190. A207- A215, A226 and A232-A233 a virtual smart card is created to access the linking engine. At line A207 the boot file is imported to the application as an argument. The boot argument is then used at line A211 in creating the profile instance of the smart card. At line A237-A238 the process of checking the boot find the appropriate dictionary is initiated. At lines A264 and A265 the dictionary determined to be the appropriate dictionary, as described above in connection with **Figure 6**, is used as the application protocol. Once the dictionary used as the application protocol is specified, the application logic unit is then free to implement the logic of the application using the hardware specific information and rules specified in the dictionary. Accordingly, at

line A288 the application begins the logic of implementing a loyalty application. The logic of implementing the loyalty application continues through to line A773. The logic of the application includes security check methods, balance check methods, methods to redeem and award loyalty points, as well as methods to report and collect information form the cardholder and the merchant sponsoring the loyalty program

For example, from line A288 through line A360 the application logic unit provides a security check procedure.

As an additional example, the application logic unit reads the value of the transaction for crediting loyalty points to the cardholder from line A367 through line A375. The application logic unit prompts the entering of the amount of the transaction at lines A370-A372. The amount of the transaction is read at line A374.

The example application logic unit contains four verbs: GetCardInfo, verify, TransactionAward, and TransactionRedeem.

The first method call for the verb GetCardInfo is at line A410. The first method call for the verb verify is at line A657. The first method call for the verb TransactionAward is at line A533. The first method call for the verb TransactionRedeem is at line A576. The first method call for all four of the verbs used in the application logic unit are after the application logic unit has specified the dictionary to be used as the application protocol. In this manner the definitions of the verbs in the application protocol are linked to the method in the application logic unit prior to the invocation of the method call.

```
A1:   package applications.loyalty;
A2:
A3:   import applications.loyalty.crypto.*;
A4:
A5:   import com.gemplus.scidl.framework.*;
A6:   import com.gemplus.scidl.util.*;
A7:   import com.gemplus.scidlx.framework.*;
A8:
A9:   import java.awt.event.*;
A10:
A11:  import java.util.Enumeration;
A12:  import java.util.Calendar;
```

```
A13:  import java.util.StringTokenizer;
A14:
A15:  public class LoyaltyInterface {
A16:
A17:    // class members
A18:    private static final String welcome =
A19:    "\nWELCOME to the Loyalty application (c) Solves inc." +
A20:    "\nThis application manages a Loyalty program on two"+
A21:    "\nGemplus smart cards:" +
A22:    "\n   * GemClub (Gemplus Loyalty card)" +
A23:    "\n   * GemXplore (Gemplus GSM card)" +
A24:    "\nEach card interface has been described using SCIDL."+
A25:    "\nThrough this interface, the same application can communicate" +
A26:    "\nwith multiple smart cards."+
A27:    "\n \nSource code and XML descriptions are available at:" +
A28:    "\nhttp:\\\\gemplus.com.\\solves\\applications" +
A29:    "\n \n** PLEASE INTRODUCE YOUR CARD TO START THE APPLICATION **";
A30:
A31:    String xml_filename;                    // this is the xml file to use
A32:    // <MODIFICATION FOR THE INTERFACE>
A33:    // PREVIOUS IMPLEMENTATION
A34:    // RunProcess virtualCard               // the virtual card instance
A35:    // NEW IMPLEMENTATION
A36:    RunProcess processEngine;               // the process engine instance
A37:    // END OF MODIFICATION
A38:    int iAmount = 0;                        // the transaction amount
A39:    PosTerminal theTerminal;                // the terminal instance
A40:    PosPinpad pinpad;                       // the terminal's pinpad
A41:    Object[] responseProcess = null;        // the process' responses
A42:    Argument[] arguments = null;            // the Argument container
A43:
A44:    long serialNumber = 0;                  // the card serial number
A45:    String cardHolderInfo = "";             // the card user information
A46:    int lastTransactionDate = 0;            // the last transaction date
A47:    String lastTransactionInfo = "";        // the last transaction
A48:    int lastTransactionAmount = 0;          // the last transaction amount
A49:    int cardBalance = 0;                    // the card balance
A50:    int transactionType = 0;                // the transaction type
A51:    Calendar date;                          // the date instance
A52:
A53:    int state = 0;                          // the state machine index
A54:    // <MODIFICATION FOR THE INTERFACE>
A55:    VirtualCardImpl virtualCard;            // the virtual card
A56:    // END OF MODIFICATION
A57:
A58:    // state machine index
A59:    public static final int NONE_STATE = 0;
A60:    public static final int AWARD_STATE = 1;
A61:    public static final int REDEEM_STATE = 2;
A62:    public static final int CHV_VERIFICATION = 4;
A63:
A64:    public static final int AWARD_TRANSACTION = 1;
A65:    public static final int REDEEM_TRANSACTION = 2;
A66:
A67:    /**
A68:     * Class contructor
A69:     *
A70:    */
```

```
A71:    public LoyaltyInterface() { }
A72:
A73:
A74:    // this is the class back implementation using the plugin interface
A75:    PlugIn plugIn = new PlugInAdapter() {
A76:      private Argument[] theArguments = null;
A77:        private Object[] r = null;
A78:
A79:        // ************************************************************
A80:        // *********** Implement the call back methods ***************
A81:        // ************************************************************
A82:        public Object[] runCallback(Object[] _o, String _message) {
A83:          // _o[1] returns the arguments of the process
A84:          // _o[0] returns the member holder of the process for th
previous steps
A85:
A86:          // for debug only
A87:    MemberHolder memberHolder = (MemberHolder) _o[0];
A88:    System.out.println("this is the list of member" + memberHolder);
A89:    for(Enumeration localMembers = memberHolder.getMemberList();
A90:          localMembers.hasMoreElements(); )
A91:       System.out.println("this is the member found in the callback: "
+
A92:                          ((Member) localMembers.nextElement()).getName());
A93:    // end of debug
A94:
A95:    // this is the call back to get the related XML file according
A96:    // to the smart card inserted in the reader
A97:    if(_message.substring(0, 5).equals("file:")) {
A98:        runApplication(_message);
A99:        r = _o; }
A100:
A101:   // this is the certificate process callback from the XML
A102:   else if(_message.substring(0, 6).equals("crypto")) {
A103:   theArguments = runTransactionCertificate(_o, _message);
A104:   r = _o; }
A105:
A106:   // this is the balance management callback from the XML
A107:   else if(_message.substring(0, 12).equals("awardBalance")) {
A108:       NotSecuredPurse localPurse = NotSecuredPurse.getInstance();
A109:
A110:       MemberHolder outMembers = (MemberHolder) _o[0];
A111:       Object[] arguments = (Object[]) _o[1];
A112:
A113:       // here we can use reflexion
A114:       localPurse.setBalance(outMembers, cardBalance);
A115:
A116:       MemberHolder responseArgs = localPurse.award(outMembers,
A117:iAmount);
A118:
A119:       Object[] plugInArgs = new Object[2];
A120:       plugInArgs[0] = responseArgs;
A121:       plugInArgs[1] = arguments;
A122:
A123:       r = plugInArgs; }
A124:
A125:   else if(_message.substring(0, 13).equals("redeemBalance")) {
A126:       NotSecuredPurse localPurse = NotSecuredPurse.getInstance();
```

```
A127:
A128:       MemberHolder outMembers = (MemberHolder) _o[0];
A129:       Object[] arguments = (Object[]) _o[1];
A130:
A131:       // here we can use reflexion instead
A132:       localPurse.setBalance(outMembers, cardBalance);
A133:
A134:       MemberHolder responseArgs = localPurse.redeem(outMembers,
A135iAmount);
A136:
A137:       Object[] plugInArgs = new Object[2];
A138:       plugInArgs[0] = responseArgs;
A139:       plugInArgs[1] = arguments;
A140:
A141:       r = plugInArgs; }
A142:  return r; }
A143:
A144: // ************************************************************
A145: // *********** Implement the plug in controls ***************
A146: // ************************************************************
A147:  public ResponseApdu runPlugIn(Object[] _o, String _message) {
A148:   // _o[1] returns the arguments of the process
A149:   // _o[0] returns the member holder of the previous steps
A150:
A151:   // for debug only
A152:   System.out.println("this is the list of member");
A153:   MemberHolder memberHolder = (MemberHolder) _o[0];
A154:   for(Enumeration localMembers = memberHolder.getMemberList();
A155:         localMembers.hasMoreElements(); )
A156:       System.out.println("this is the member found in the plugin: " +
A157:                          ((Member)
localMembers.nextElement()).getName());
A158:   // end of debug
A159:
A160:   // get the content message
A161:   StringTokenizer st = new StringTokenizer(_message, "?");
A162:   String messageId = st.nextToken().trim();
A163:
A164:   String message = st.nextToken().trim();
A165:   st = new StringTokenizer(message, "=");
A166:   String itemName = st.nextToken().trim();
A167:   String itemValue = st.nextToken().trim();
A168:
A169:if(messageId.equals("pluginMessage") && itemName.equals("secretCode"))
{
A170: theTerminal.displayMessage("ACCESS PROTECTED" +
A171:        "\n \nThe user password is required to start the application."
+
A172:                "\nCould you enter your password using the keypad" +
A173:                "\n and press Valid key for verification.");
A174:       state |= CHV_VERIFICATION ;
A175:       pinpad.readMessage();   // start reading message from the
pinpad
A176:
A177:       while((state & 0x04) != 0) {
A178:         try { Thread.sleep(1000); }
A179:         catch(Exception e) { } } } // this wait till the present
A180:passwork is completed
```

```
A181:    return null; }
A182:   };
A183:
A184:
A185:   /**
A186:    * main class
A187:    *
A188:    * @param _args An array of strings to get the application
parameters
A189:   */
A190:   public static void main(String[] args) {
A191: LoyaltyInterface application = new LoyaltyInterface();
A192: application.start(args);
A193:   }
A194:
A195:
A196:   /**
A197:    * The method to run the application
A198:    *
A199:   */
A200:   private void start(String[] args) {
A201: date = Calendar.getInstance();
A202: pinpad = new PosPinpad(this.listener());
A203: theTerminal = new PosTerminal(pinpad);
A204: theTerminal.displayMessage(welcome);
A205:
A206: // create the virtual smart card to access the engine
A207: xml_filename = "file:////" + args[0];
A208: System.out.println("This is the XML file open: " + xml_filename);
A209: // create the profile instance that contains all the smart card
A210:profiles
A211: ProfileImpl theProfiles = new ProfileImpl(xml_filename);
A212: // get the content of the xml file
A213: // note ** the next version of the engine should remove that
invocation
A214: theProfiles.setProfiles();
A215: Enumeration profileList = theProfiles.getProfiles();
A216:
A217: // create the process helper for the call back
A218: CallBackList callbackList = new CallBackList();
A219: callbackList.addCallback(plugIn);
A220: // create the process helper for the plugIn
A221: PluginList pluginList = new PluginList();
A222: pluginList.addPlugin(plugIn);
A223:
A224:
A225: // create the process helper --- OCF 1.1
A226: Terminal OcfLayer = new OcfTerminal11();
A227: // <MODIFICATION FOR THE INTERFACE>
A228: // PREVIOUS IMPLEMENTATION
A229: // virtualCard = new RunProcessImpl(OcfLayer, pluginList,
A230:callbackList);
A231: // NEW IMPLEMENTATION
A232: processEngine = new RunProcessImpl(OcfLayer, pluginList,
callbackList);
A233: virtualCard = new VirtualCardImpl(processEngine, theProfiles);
A234: // END OF MODIFICATION
A235:
```

```
A236: // get the full file name of the application SCIDL file
A237: try { responseProcess = virtualCard.runProcess("getFile", arguments,
A238:                                          theProfiles); }
A239:
A240: catch(ProfileException pe) {
A241:   theTerminal.displayMessage("The system has detected a problem." +
A242:                                 "\nError message: " + pe.getMessage() +
A243:                     "\n \nPlease, contact your service provider."); }
A244: catch(RunProcessException rpe) {
A245:   theTerminal.displayMessage("The runtime environment has detected a
A246:problem." +
A247:                                 "\nError message: " + rpe.getMessage() +
A248:                     "\n \nPlease, contact your service provider."); }
A249:
A250:  }
A251:
A252:  /**
A253:   * The application implementation
A254:   *
A255:   * @param _file A string that represents the full path of the XML
file
A256:   * to use as the application protocol
A257:   * @return An array of argument that may be null becaus the engine
A258:doesn't
A259:   * expect any response
A260:  */
A261:  private void runApplication(String _file) {
A262:
A263: // set te new framework
A264: ProfileImpl theProfiles = new ProfileImpl(_file);
A265: theProfiles.setProfiles();
A266:
A267: // there is only one description in the XML file
A268: CardProfile profile =
A269:   (CardProfile) theProfiles.getProfiles().nextElement();
A270: // <MODIFICATION FOR THE INTERFACE>
A271: // PREVIOUS IMPLEMENTATION
A272: //    virtualCard.setProfile(profile);
A273: // NEW IMPLEMENTATION
A274: processEngine.setProfile(profile);
A275: // END OF MODIFICATION
A276:
A277: runDisplay();
A278:  }
A279:
A280: /**
A281:   * The method runs the generation certification for the given
A282:   * smart card inserted in the reader
A283:   *
A284:   * @param _message A String that represents the message received
from
A285:   * the XML file during the call back event.
A286:   * @return An array of arguments that represents the data process
A287:  */
A288:  public Argument[] runTransactionCertificate(Object[] _o, String
A289:_message) {
A290: Argument[] arguments = null;
A291:
```

```
A292: // get the algorithm by checking the message (_message)
A293: // the message content has the form of: "crypto?award=DES"
A294: StringTokenizer st = new StringTokenizer(_message, "=");
A295: String algo = st.nextToken().trim(); algo = st.nextToken().trim();
A296:
A297: if(algo.equals("DES")) {
A298:    arguments = (Argument[]) _o[1];        // get the arguments
received A299:from the callback
A300:    int iTransaction = 0;
A301:    byte[] baTransaction =
A302:  { 'P', 'i', 'z', 'z', 'a', '#', '1', ' ', ' ', ' ', ' ', ' ', ' ',
' A303:', ' ', ' ' };
A304:
A305:    byte[] baCertificate = new byte[8];
A306:    Argument transactionAtt = null;
A307:
A308:    // get the transaction argument
A309:    for(int i = 0; i < arguments.length; i++) {
A310:        if(arguments[i].getName().equals("transaction")) {
A311:          transactionAtt = arguments[i];
A312:          iTransaction = i; } }
A313:
A314:    DES desAlgorithm = new DES();
A315:
A316:    // both key are iitialized with zero value, this is for the initial
A317:    // implementation because the system doesn't support crypto
A318:    // The next version of the implemention will carry real keys
A319:    byte[] baAwardKey = { 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07,
0x08 A320:};
A321:    byte[] baRedeemKey = { 0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x17,
0x18 A322};
A323:    DesKey awardKey = new DesKey(baAwardKey);
A324:    DesKey redeemKey = new DesKey(baRedeemKey);
A325:
A326:    // create the certification data in
A327:    baCertificate[0] = (byte) (date.get(Calendar.YEAR) - 1900);
A328:    baCertificate[1] = (byte) (date.get(Calendar.MONTH) + 1);
A329:    baCertificate[2] = (byte) date.get(Calendar.DAY_OF_MONTH);
A330:    baCertificate[3] = (byte) date.get(Calendar.HOUR_OF_DAY);
A331:    baCertificate[4] = (byte) date.get(Calendar.MINUTE);
A332:    baCertificate[5] = (byte) Hex.integerToBytes(iAmount)[1];
A333:    baCertificate[6] = (byte) Hex.integerToBytes(iAmount)[2];
A334:    baCertificate[7] = (byte) Hex.integerToBytes(iAmount)[3];
A335:
A336:    if(state == AWARD_STATE) {
A337:        try {
A338:          desAlgorithm.setkey(awardKey.getKey());
A339:          desAlgorithm.encrypt(baCertificate);
A340:          System.arraycopy(baCertificate, 0, baTransaction, 8, 8); }
A341:        catch(CryptoException e) {
A342:          System.out.println("this is cryptoException: " +
A343:e.getMessage()); }
A344:        Variable transaction = ((Variable)
A345:arguments[iTransaction].getMember());
A346:        transaction.setValue(baTransaction); }
A347:
A348:    if(state == REDEEM_STATE) {
A349:        try {
```

```
A350:          desAlgorithm.setkey(redeemKey.getKey());
A351:          desAlgorithm.encrypt(baCertificate);
A352:          System.arraycopy(baCertificate, 0, baTransaction, 8, 8); }
A353:        catch(CryptoException e) {
A354:          System.out.println("this is cryptoException: " +
A355:e.getMessage()); }
A356:        Variable transaction = ((Variable)
A356:arguments[iTransaction].getMember());
A357:        transaction.setValue(baTransaction); } }
A358:
A359:    return arguments;
A360:   }
A361:
A362:
A363:   /**
A364:    * The method runs the award transaction
A365:    *
A366:   */
A367:   public void runAward() {
A368:
A369: if(state == NONE_STATE) {
A370:    theTerminal.displayMessage("AWARD TRANSACTION"+
A371:                    "\n \nEnter the transaction amount using the
keypad"+
A372:                     "\nand press Valid key for confirmation.");
A373:    state = AWARD_STATE;
A374:    pinpad.readMessage(); } // start reading message from the pinpad
A375:   }
A376:
A377:   /**
A378:    * The method runs the redeem transaction
A379:    *
A380:   */
A381:   public void runRedeem() {
A382: if(state == NONE_STATE) {
A383:    theTerminal.displayMessage("REDEEM TRANSACTION"+
A384:              "\n \nEnter the transaction amount using the keypad"+
A385:                          "\nand press Valid key for confirmation.");
A386:    state = REDEEM_STATE;
A387:    pinpad.readMessage(); }  // start reading message from the pinpad
A388:   }
A389:
A390:
A391:   /**
A392:    * The method runs the redeem transaction
A393:    *
A394:   */
A395:   public void runCancel() {
A396: state = NONE_STATE;
A397: theTerminal.displayMessage("ABORTED TRANSACTION"+
A398:           "\n \nThe transaction process has been aborted by the
user.");
A399:   }
A400:
A401:
A402:   /**
A403:    * The method runs the redeem transaction
A404:    *
```

```
A405:  */
A405:  public void runDisplay() {
A407: Argument[] nullArgument = null;
A408:
A409: // get the card information
A410: try { responseProcess = virtualCard.GetCardInfo(nullArgument); }
A411: // <MODIFICATION FOR THE INTERFACE>
A412: // PREVIOUS IMPLEMENTATION
A413: // responseProcess = virtualCard.runProcess("GetCardInfo",
A414nullArgument);
A415: // Note for the developer: The previous implementation remains valid
A416:because the
A417: // converter creates the relevant method to wrap this invocation. It
is A418:worth noticing
A419: // this manner of invoking the smart card process becomes depreciated
A420:when using the
A421: // interface implementation provided by the Converter.
A422: //
A423: // The rest of the implementation remains depreciated and doesn't use
A424:the method
A425: // invocation on purpose. The target of this implementation is to
A426:measure the
A427: // correction to bring in the loyalty.java file to move from one
A428:implementation to
A429: // the stub implementation.
A430:
A431: catch(ProfileException pe) {
A432:   theTerminal.displayMessage("The system has detected a problem." +
A433:                              "\nError message: " + pe.getMessage() +
A434:                    "\n \nPlease, contact your service provider."); }
A435: catch(RunProcessException rpe) {
A436:   theTerminal.displayMessage("The runtime environment has detected a
A437:problem." +
A438:                      "\nError message: " + rpe.getMessage() +
A439:                    "\n \nPlease, contact your service provider."); }
A440:
A441:
A442: serialNumber = ((Variable) findMember("CardSerialNumber",
A443:                                responseProcess)).toInteger();
A444:
A445: cardHolderInfo = ((Variable) findMember("CardHolderName",
A446:                               responseProcess)).toString();
A447: cardHolderInfo.trim();
A448:
A449: lastTransactionInfo = ((Variable) findMember("LastTransactionInfo",
A450:                                     responseProcess)).toString();
A451:
A452: lastTransactionAmount = ((Variable)
findMember("LastTransactionAmount",
A453                                      responseProcess)).toInteger();
A454:
A455: lastTransactionDate = ((Variable) findMember("LastTransactionDate",
A456:                                     responseProcess)).toInteger();
A457:
A458: cardBalance = ((Variable) findMember("cardBalance",
A459:                               responseProcess)).toInteger();
A460:
A461: transactionType = ((Variable) findMember("transactionType",
```

```
A462:                                   responseProcess)).toInteger();
A463:
A464: if(transactionType == 1) {
A465:   theTerminal.displayMessage("Welcome to Solves E-commerce server." +
A466:                   "\n \nSmart card ID: " + serialNumber +
A467:                   "\nUsername: " + cardHolderInfo +
A468:                   "\n \nThis is your account information..." +
A469:                   "\nLast award transaction: " + lastTransactionInfo
+
A470:                   "\nDate: " +
A471:                    (byte) (lastTransactionDate) + "-" +
A472:                    (byte) (lastTransactionDate >> 8) + "-" +
A473:                    (byte) (lastTransactionDate >> 16) +
A474:                    "\nAmount: " + lastTransactionAmount +
A475:                    "\n \nYour current balance is: " + cardBalance +
A476:                    " point(s)"); }
A477: else if(transactionType == 2) {
A478:         theTerminal.displayMessage("Welcome to Solves E-commerce
A479:server." +
A480:                   "\n \nSmart card ID: " + serialNumber +
A481:                   "\nUsername: " + cardHolderInfo +
A482:                   "\n \nThis is your account information..." +
A483:                   "\nLast redeem transaction: " + lastTransactionInfo
+
A484:                   "\nDate: " +
A485:                    (byte) (lastTransactionDate) + "-" +
A486                     (byte) (lastTransactionDate >> 8) + "-" +
A487:                    (byte) (lastTransactionDate >> 16) +
A488:                    "\nAmount: " + lastTransactionAmount +
A489:                    "\n\nYour current balance is: " + cardBalance +
A490:                    " point(s)"); }
A491: else {
A492:   System.out.println("This is the card holder name: " +
A493:cardHolderInfo);
A494:   theTerminal.displayMessage("Welcome to Solves E-commerce server." +
A495:                   "\n \nSmart card ID: " + serialNumber +
A496:                   "\nUsername: " + cardHolderInfo +
A497:                   "\n \nYour current balance is: " + cardBalance +
A498:                    " point(s)"); }
A499:
A500:  }
A501:
A502:
A503:  /**
A504:   * The method runs the redeem transaction
A505:   *
A506:  */
A507:  public void runBonus() {
A508: System.out.println("TerminalPinPad this is the key press: bonus");
A509:
A510: // set the transaction date
A511: byte[] baDate = { (byte) (date.get(Calendar.YEAR) - 1900),
A512:                            (byte) (date.get(Calendar.MONTH) + 1),
A513:                            (byte) date.get(Calendar.DAY_OF_MONTH) };
A514:
A515: if(state == AWARD_STATE) {
A516:// start the transaction processing
A517:// 1 - update the amount to increase
```

```
A518:byte[] baAmount = new byte[3];
A519:
A520:iAmount = Integer.parseInt(theTerminal.getMessage());
A521:baAmount[0] = Hex.integerToBytes(iAmount)[1];
A522:  baAmount[1] = Hex.integerToBytes(iAmount)[2];
A523:  baAmount[2] = Hex.integerToBytes(iAmount)[3];
A524:
A525:  // 2 - prepare the transaction
A526:  Argument amountAtt = new Argument("amount", baAmount);
A527:  Argument dateAtt = new Argument("date", baDate);
A528:  byte[] tid = new byte[16];
A529:  Argument terminalId = new Argument("transaction", tid);
A530:  Argument[] awardArguments = { amountAtt, dateAtt, terminalId };
A531:
A532:  // 3 - run the transaction
A533:  try { responseProcess = virtualCard.runProcess("TransactionAward",
A534:                                          awardArguments); }
A535:  catch(ProfileException pe) {
A536:    theTerminal.displayMessage("The system has detected a problem." +
A537:                    "\nError message: " + pe.getMessage() +
A538:                    "\n \nPlease, contact your service povider."); }
A539:  catch(RunProcessException rpe) {
A540:    theTerminal.displayMessage("The runtime environment has detected
a A541:problem." +
A542:                            "\nError message: " + rpe.getMessage() +
A543:                    "\n \nPlease, contact your service provider."); }
A544:
A545:  int newBalance = ((Variable) findMember("newBalance",
A546:                                  responseProcess)).toInteger();
A547:  System.out.println("This is the new balance... : " + newBalance);
A548:  theTerminal.displayMessage("PROCESSING AWARD TRANSACTION..." +
A549:                          "\n \nThe number of point(s) to award was: "
+
A550:                          theTerminal.getMessage() +
A551:                          "\n \nAward transaction successful." +
A552:                          "\nThe new balance of the card is: " +
A553:                          newBalance);
A554:  state = NONE_STATE; }  // stop the state machine
A555:
A556: if(state == REDEEM_STATE) {
A557:
A558:  // start the transaction processing
A559:  // 1 - update the amount to redeem
A560:  byte[] baAmount = new byte[3];
A561:
A562:  iAmount = Integer.parseInt(theTerminal.getMessage());
A563:  baAmount[0] = Hex.integerToBytes(iAmount)[1];
A564:  baAmount[1] = Hex.integerToBytes(iAmount)[2];
A565:  baAmount[2] = Hex.integerToBytes(iAmount)[3];
A566:
A567:  System.out.println("this is the redeem bonus press");
A568:
A569:  // 2 - prepare the transaction
A570:  Argument amountAtt = new Argument("amount", baAmount);
A571:  Argument dateAtt = new Argument("date", baDate);
A572:  byte[] tid = new byte[16];
A573:  Argument terminalId = new Argument("transaction", tid);
A574:  Argument[] awardArguments = { amountAtt, dateAtt, terminalId };
```

```
A575:
A576:    try { responseProcess = virtualCard.runProcess("TransactionRedeem",
A577:
        A578:awardArguments); }
A579:    catch(ProfileException pe) {
A580:      theTerminal.displayMessage("The system has detected a problem." +
A581:                              "\nError message: " + pe.getMessage() +
A582:                         "\n \nPlease, contact your service provider."); }
A583:    catch(RunProcessException rpe) {
A584:      theTerminal.displayMessage("The runtime environment has detected
a A585:problem." +
A586:                                    "\nError message: " + rpe.getMessage() +
A587:                         "\n \nPlease, contact your service provider."); }
A588:
A589:    int newBalance = ((Variable) findMember("newBalance",
A590:                              responseProcess)).toInteger();
A591:    System.out.println("This is the new balance: " + newBalance);
A592:    theTerminal.displayMessage("PROCESSING REDEEM TRANSACTION..." +
A593:                         "\n \nThe number of point(s) to redeem was: " +
A594:                         theTerminal.getMessage() +
A595:                         "\n \nRedeem transaction successful." +
A596:                         "\nThe new balance of the card is: " +
A597:                         newBalance);
A599:    state = NONE_STATE; }  // stop the state machine
A600:
A601:  }
A602:
A603:
A604:  /**
A605:   * The method runs the redeem transaction
A606:   *
A607:  */
A608:  public void runValid() {
A609:
A610: if(state == AWARD_STATE) {
A611:    pinpad.readMessage();          // stop reading the message from
pinpad
A612:    theTerminal.displayMessage("PROCESSING AWARD TRANSACTION..." +
A613:                                    "\n \nThe transaction amount is: " +
A614:                                    theTerminal.getMessage() + " point(s)"
+
A615:     "\n \nPlease press Bonus key to confirm or Cancel to abort."); }
A616:
A617: if(state == REDEEM_STATE) {
A618:    pinpad.readMessage();          // stop reading the message from
pinpad
A619:    theTerminal.displayMessage("PROCESSING REDEEM TRANSACTION..." +
A620:                         "\n \nThe transaction amount is: " +
A621:                         theTerminal.getMessage() + " point(s)" +
A622:  "\n \nPlease press Bonus key to confirm or Cancel to abort.");    }
A623:
A624: if((state & 0x04) == CHV_VERIFICATION) {
A625:    pinpad.readMessage();          // stop reading the message from
pinpad
A626:    theTerminal.displayMessage("PASSWORD VERIFICATION...." +
A627:          "\n \nPassword entered by user:" + theTerminal.getMessage());
A628:
A629:    // real present the related password
```

```
A630:    byte[] baValuex = new byte[8];
A631:    String pincodeLiteral = theTerminal.getMessage();
A632:    char[] messageReceived = pincodeLiteral.toCharArray();
A634:    byte[] baMessageReceived = new byte[messageReceived.length];
A635:
A636:    for(int i = 0; i < messageReceived.length; i++)       // conversion
A637:from char to byte
A638:        baMessageReceived[i] = (byte) messageReceived[i];
A639:
A640:    for(int i = 0; i < baValuex.length; i++)           // set the
A641:default value
A642:        baValuex[i] = (byte) 0xff;
A643:
A644:    System.arraycopy(baMessageReceived, 0,              // update the
A645:password buffer
A646:        baValuex, 0, messageReceived.length);
A647:
A648:   byte[] baValue =                                    // feak
presentation
A649:    { '1', '2', '3', '4', (byte) 0xff, (byte) 0xff, (byte) 0xff, (byte)
A650: 0xff };
A651:
A652:    Argument value = new Argument("value", baValue);  // change baValue
A653:by baValuex here
A654:    Argument index = new Argument("index", (byte) 1); // to use the
real A655:presentation
A656:    Argument[] chvArguments = { value, index };
A657:    try { responseProcess = virtualCard.runProcess("verify",
A658:chvArguments); }
A659:
A660:    catch(ProfileException pe) {
A661:      theTerminal.displayMessage("The system has detected a problem." +
A662:                                 "\nError message: " + pe.getMessage() +
A663:                      "\n \nPlease, contact your service provider."); }
A664:    catch(RunProcessException rpe) {
A665:      theTerminal.displayMessage("The runtime environment has detected
a A666:problem." +
A667:                                 "\nError message: " + rpe.getMessage() +
A668:                      "\n \nPlease, contact your service provider."); }
A669:
A670:    state = state & 0xFB;         // reset the present password state
A671:machine
A672:    System.out.println("this is the state machine in valid: " + state);
}
A673:  }
A674:
A675:
A676:  // ************************************************************
A677:  // *********** Create the generic control listeners **********
A678:  // ************************************************************
A679:  final ActionListener listener() {
A680: ActionListener theListener = new ActionListener() {
A681:   public void actionPerformed(ActionEvent _e) {
A682:       PosButton button = (PosButton) _e.getSource();
A683:       if(button.getText().equals("award")) runAward();
A684:       if(button.getText().equals("valid")) runValid();
A685:       if(button.getText().equals("display")) runDisplay();
A686:       if(button.getText().equals("bonus")) runBonus();
```

```
A687:       if(button.getText().equals("cancel")) runCancel();
A687:       if(button.getText().equals("redeem")) runRedeem(); } };
A688: return theListener;
A689:  }
A690:
A691:
A692:  /**
A693:   * The method looks up in the response members and returns the
related A694:member
A695:   *
A696:   * @param _member A string that represents the member in the
responses
A697:   * @return A member instance that represents the member found in the
A698:response list,
A699:   * otherwise the method reutrns null
A700:  */
A701:  private Member findMember(String _member, Object[] _responseProcess)
{
A702: Object[] localResponses = _responseProcess;
A703: Member foundMember = null;
A704:
A705: for(int i = 0; i < localResponses.length; i++) {
A706:   ResponseApdu ra = (ResponseApdu) localResponses[i];
A707:
A707:   MemberHolder responseMembers = ra.getResponseMembers();
A708:   Enumeration members = responseMembers.getMemberList();
A709:   while(members.hasMoreElements()) {
A710:       Member member = (Member) members.nextElement();
A711:       if(member.getName().equals(_member))
A712:         foundMember = member; } }
A713: return foundMember;
A714:  }
A715:}
A716:
A717:class NotSecuredPurse {
A718:
A719:  public void NotSecuredPurse() { }
A720:
A721:  private static NotSecuredPurse localPurse = null;
A722:  private int iBalance;
A723:  private boolean zBalance = false;
A724:
A725:  public MemberHolder award(MemberHolder _o, int _iAmount) {
A726: MemberHolder localMember = _o;
A727:
A728: Member newBalance = localMember.getMember("newBalance");
A729: iBalance += _iAmount;
A730: byte[] baAmount = new byte[3];
A731: baAmount[0] = Hex.integerToBytes(iBalance)[1];
A732: baAmount[1] = Hex.integerToBytes(iBalance)[2];
A733: baAmount[2] = Hex.integerToBytes(iBalance)[3];
A734: ((Variable) newBalance).setValue(baAmount);
A735: return localMember; }
A736:
A737:  public MemberHolder redeem(MemberHolder _o, int _iAmount) {
A738: MemberHolder localMember = _o;
A739:
A740: Member newBalance = localMember.getMember("newBalance");
```

```
A741: iBalance -= _iAmount;
A742: byte[] baAmount = new byte[3];
A743: baAmount[0] = Hex.integerToBytes(iBalance)[1];
A744: baAmount[1] = Hex.integerToBytes(iBalance)[2];
A745: baAmount[2] = Hex.integerToBytes(iBalance)[3];
A746: ((Variable) newBalance).setValue(baAmount);
A747: return localMember; }
A748:
A749:  public void setBalance(MemberHolder _o, int _iBalance) {
A750: if(!zBalance) {
A751:   iBalance = _iBalance;
A752:
A753:   MemberHolder localMemberHolder = _o;
A754:
A755:   if(localMemberHolder.isInMemberList("newBalance")) {
A756:       Member newBalance = localMemberHolder.getMember("newBalance");
A757:       byte[] baAmount = new byte[3];
A758:       baAmount[0] = Hex.integerToBytes(iBalance)[1];
A759:       baAmount[1] = Hex.integerToBytes(iBalance)[2];
A760:       baAmount[2] = Hex.integerToBytes(iBalance)[3];
A761:       ((Variable) newBalance).setValue(baAmount); }
A762:   zBalance = true; }
A763:  }
A763:
A765:  public int getBalance() { return iBalance; }
A766:
A767:  public static NotSecuredPurse getInstance() {
A768: if(localPurse == null)
A769:   localPurse = new NotSecuredPurse();
A770:
A771: return localPurse;
A772:  }
A773:}
```

APPLICATION PROTCOL

The example of an application protocol listed below from line B1 through B803 provides the data and rules for implementing the logic of the application logic unit listed above on a Gemplus™ GemXplore98™ smart card. The card the dictionary is compatible with is specified at line B6. This example dictionary includes definitions on four verbs: GetCardInfo, verify, TransactionAward, and TransactionRedeem.

At line B10 the verb GetCardInfo is specified. The two method calls for the verb GetCardInfo in the example of the application logic unit, described above, are at lines A410 and A413. The definition of the verb GetCardInfo is provided at lines B11 through B293. Lines B15 through B57 list data for the implementation of the verb GetCardInfo. Lines B59

through B293 list the rules used in implementing the verb GetCardInfo on the Gemplus GemXplore98 smart card.

At line B295 the verb verify is specified. The definition corresponding to the verb verify is provided at lines B295 through B326. Lines B296 through B302 list data for the implementation of the verb verify. Lines B303 through B326 list the rules used in implementing the verb verify on the Gemplus GemXplore98 smart card. The only method call for the verb verify in the above example of an application logic unit is at line A657.

At line B328 the verb TransactionAward is specified. The definition of the verb TransactionAward is provided at lines B328 through B554. Lines B332 through B377 list data for the implementation of the verb TransactionAward. Lines B379 through B554 list the rules used in implementing the verb TransactionAward on the Gemplus GemXplore98 smart card. The only method call for the verb TransactionAward in the above example of an application logic unit is at line A533.

At line B555 the verb TransactionRedeem is specified. The definition of the verb TransactionRedeem is provided at lines B555 through B785. Lines B559 through B595 list data for the implementation of the verb TransactionRedeem. Lines B597 through B803 list the rules used in implementing the verb TransactionRedeem on the Gemplus GemXplore98 smart card. Lines B597 through B785 list the rules used in implementing the verb TransactionRedeem on the Gemplus GemXplore98 smart card. The only method call for the verb TransactionRedeem in the above example of an application logic unit is at line A576.

In the definitions of the four verbs contained in the application protocol, the data and rules used to implement the definition of the verb is particular to the smart device, in this example a Gemplus GemXplore98 smart card. Definitions for the same four verbs: GetCardInfo, verify, TransactionAward, and TransactionRedeem, for implementation with another smart device in connection with the example application logic unit would typically have definitions that differed from the example definitions presented. The definitions could differ in either the data, the rules, or both.

```
B1:<?xml version = "1.0" ?>
B2:<!DOCTYPE CardDocument SYSTEM "..\dtd\com\gemplus\xml\SCIDL.dtd">
B3:
B4:
B5:<CardDocument>
B6:<Profile Type = "GemXplore98" Version = "0.1">
B7:    <Signature Type = "Card">0x80 0x69 0xAF 0x03 0x07 0x03 0x52 0x00 0x00
B8:0x0A 0x0E 0x83 0x3E 0x9F 0x16</Signature>
B9:
B10:    <Process Name = "GetCardInfo">
B11:    <Doc>The process represents the manner to retreive the card
B12:information</Doc>
B13:    <Doc>The members returned are not listed</Doc>
B14:
B15:        <Variable Name = "cardSerialNumberFile" Type = "byteArray"
B16:Default = "0x2F 0xE2"/>
B17:        <Variable Name = "RFU_1" Type = "byteArray" Default = "0x00
B18:0x00"/>
B19:        <Variable Name = "memoryLeft" Type = "byteArray" Default =
"0x00 B20:0x00"/>
B21:        <Variable Name = "fileId" Type = "byteArray" Default = "0x00
B22:0x00"/>
B23:        <Variable Name = "fileType" Type = "byte" Default = "0x00"/>
B24:        <Variable Name = "fileFeatures" Type = "byteArray" Default =
B25:"0x00 0x00 0x00 0x00 0x00"/>
B26:        <Variable Name = "restOfResponse" Type = "byte" Default =
B27:"0x00"/>
B28:
B29:        <Variable Name = "index" Type = "byte" Default = "0"/>
B30:        <Variable Name = "value" Type = "byteArray" Length = "8"
Default B31:= ""/>
B32:        <Variable Name = "offset_1" Type = "byte" Default = "0x00"/>
B33:        <Variable Name = "offset_2" Type = "byte" Default = "0x00"/>
B34:        <Variable Name = "length" Type = "byte" Default = "0x08"/>
B35:        <Variable Name = "CardSerialNumber" Type = "byteArray" Length =
B36:"8" Default = ""/>
B37:
B38:        <Variable Name = "cardInformationFile" Type = "byteArray"
Default B39:= "0x2F 0x30"/>
B40:        <Variable Name = "LastTransactionInfo" Type = "byteArray"
Length B41:= "16" Default = ""/>
B42:        <Variable Name = "LastTransactionDate" Type = "byteArray"
Default B43:= "0x00 0x00 0x00"/>
B44:        <Variable Name = "LastTransactionAmount" Type = "byteArray"
B45:Default = "0x00 0x00 0x00"/>
B46:        <Variable Name = "transactionType" Type = "byte" Default =
B47:"0x00"/>
B48:
B49:        <Variable Name = "cardBalanceFile" Type = "byteArray" Default =
B50:"0x2F 0x31"/>
B51:        <Variable Name = "cardBalance" Type = "byteArray" Default =
"0x00 B52:0x00 0x00"/>
B53:
B54:        <Variable Name = "cardHolderFile" Type = "byteArray" Default =
B55:"0x2F 0x32"/>
```

```
B56:        <Variable Name = "CardHolderName" Type = "byteArray" Length =
B57:"20" Default = ""/>
B58:
B59:        <Apdu Id = "Select">
B60:              <Command>
B61:              <Header Class = "0xA0" Ins = "0xA4" P1 = "0"  P2 = "0"
Lc B62:= "2" Le = "0x20"/>
B63:              <In>cardSerialNumberFile</In>
B64:              </Command>
B65:              <Response Status = "NormalEnding" Notify = "DoNotNotify">
B66:            <Out>RFU_1</Out>
B67:            <Out>memoryLeft</Out>
B68:            <Out>fileId</Out>
B69:            <Out>fileType</Out>
B70:            <Out>fileFeatures</Out>
B71:            <Out>restOfResponse</Out></Response>
B72:              <Response Status = "MemoryProblem"><Out>"1On Error"</Out>
B73:</Response>
B74:            <Response Status = "OutOfRange"><Out> "2OnError" </Out>
B75:</Response>
B76:            <Response Status = "FileOrPatternNotFound"><Out> "@OnError"
B77:</Out> </Response>
B78:            <Response Status = "IncorrectLength"><Out> "3OnError"
</Out> B79:</Response>
B80:            <Response Status = "IncorrectP1P2"><Out> "4OnError" </Out>
B81:</Response>
B82:            <Response Status = "IncorrectClass"><Out> "5OnError" </Out>
B83:</Response>
B84:        </Apdu>
B85:
B86:        <Apdu Id = "ReadBinary">
B87:              <Command>
B88:              <Header Class = "0xA0" Ins = "0xB0" P1 = "offset_1"  P2 =
B89:"offset_2"  Le = "length"/>
B90:              <In>"Void"</In>
B91:              </Command>
B92:              <Response Status = "NormalEnding" Notify = "DoNotNotify">
B93:            <Out>CardSerialNumber</Out></Response>
B94:              <Response Status = "MemoryProblem"><Out>"Memory
B95:problem"</Out> </Response>
B96:            <Response Status = "NoneFileSelected"><Out>"File not
B97:selected"</Out> </Response>
B98:            <Response Status = "WrongTypeOfFile"><Out>"Wrong type of
B99:file"</Out> </Response>
B100:            <Response Status = "WrongAccessCondition"><Out>"Access
B101:condition"</Out> </Response>
B102:            <Response Status = "FileInvalidated"><Out> "1OnError"
</Out> B103:</Response>
B104:            <Response Status = "IncorrectP1P2"><Out> "2OnError" </Out>
B105:</Response>
B106:            <Response Status = "IncorrectClass"><Out> "3OnError"
</Out> B107:</Response>
B108:       </Apdu>
B109:
B110:       <Apdu Id = "Select">
B111:             <Command>
B112:             <Header Class = "0xA0" Ins = "0xA4" P1 = "0"  P2 = "0"
Lc B113:= "2" Le = "0x10"/>
```

```
B114:              <In> cardInformationFile</In>
B115:              </Command>
B116:              <Response Status = "NormalEnding" Notify = "DoNotNotify">
B117:             <Out>RFU_1</Out>
B118:             <Out>memoryLeft</Out>
B119:             <Out>fileId</Out>
B120:             <Out>fileType</Out>
B121:             <Out>fileFeatures</Out>
B122:             <Out>restOfResponse</Out></Response>
B123:              <Response Status = "MemoryProblem"><Out>"On Error"</Out>
B124:</Response>
B125:             <Response Status = "OutOfRange"><Out> "OnError" </Out>
B126:</Response>
B127:             <Response Status = "FileOrPatternNotFound"><Out>
"@OnError" B128:</Out> </Response>
B129:             <Response Status = "IncorrectLength"><Out> "OnError"
</Out> B130:</Response>
B131:             <Response Status = "IncorrectP1P2"><Out> "OnError" </Out>
B132:</Response>
B133:             <Response Status = "IncorrectClass"><Out> "OnError" </Out>
B133:</Response>
B134:       </Apdu>
B135:
B136:       <Apdu Id = "ReadBinary">
B137:             <Command>
B138:             <Header Class = "0xA0" Ins = "0xB0" P1 = "offset_1"  P2 =
B139:"offset_2"  Le = "50"/>
B140:             <In>"Void"</In>
B141:       </Command>
B142:       <Response Status = "NormalEnding" Notify = "DoNotNotify">
B143:             <Out>LastTransactionDate</Out>
B144:             <Out>LastTransactionAmount</Out>
B145:             <Out>transactionType</Out>
B146:             <Out>LastTransactionInfo</Out>
B147:             </Response>
B148:              <Response Status = "MemoryProblem"><Out>"On Error"</Out>
B149:</Response>
B150:             <Response Status = "NoneFileSelected"><Out> "OnError"
</Out> B151:</Response>
B152:             <Response Status = "WrongTypeOfFile"><Out> "OnError"
</Out> B153:</Response>
B154:             <Response Status = "WrongAccessCondition" Notify =
"PlugIn">
B155:             <Out>"pluginMessage?secretCode=01"</Out></Response>
B156:             <Response Status = "FileInvalidated"><Out> "OnError"
</Out> B157:</Response>
B158:             <Response Status = "IncorrectP1P2"><Out> "OnError" </Out>
B159:</Response>
B160:             <Response Status = "IncorrectClass"><Out> "OnError" </Out>
B161:</Response>
B162:       </Apdu>
B163:
B164:       <Apdu Id = "ReadBinary">
B165:             <Command>
B166:             <Header Class = "0xA0" Ins = "0xB0" P1 = "offset_1"  P2 =
B167:"offset_2"  Le = "50"/>
B168:             <In>"Void"</In>
B169:              </Command>
```

```
B170:              <Response Status = "NormalEnding" Notify = "DoNotNotify">
B171:             <Out>LastTransactionDate</Out>
B172:             <Out>LastTransactionAmount</Out>
B173:             <Out>transactionType</Out>
B174:             <Out>LastTransactionInfo</Out>
B175:             </Response>
B176:              <Response Status = "MemoryProblem"><Out>"On Error"</Out>
B177:</Response>
B178:             <Response Status = "NoneFileSelected"><Out> "OnError"
</Out> B179:</Response>
B180:             <Response Status = "WrongTypeOfFile"><Out> "OnError"
</Out> B181:</Response>
B182:             <Response Status =
B183:"WrongAccessCondition"><Out>"OnError"</Out></Response>
B184:             <Response Status = "FileInvalidated"><Out> "OnError"
</Out> B185:</Response>
B186:             <Response Status = "IncorrectP1P2"><Out> "OnError" </Out>
B187:</Response>
B188:             <Response Status = "IncorrectClass"><Out> "OnError" </Out>
B189:</Response>
B190:       </Apdu>
B191:
B192:       <Apdu Id = "Select">
B193:             <Command>
B194              <Header Class = "0xA0" Ins = "0xA4" P1 = "0"  P2 = "0"
Lc B195:= "2" Le = "0x10"/>
B196:             <In>cardBalanceFile</In>
B197:             </Command>
B198:             <Response Status = "NormalEnding" Notify = "DoNotNotify">
B199:             <Out>RFU_1</Out>
B200:             <Out>memoryLeft</Out>
B201:             <Out>fileId</Out>
B202:             <Out>fileType</Out>
B203:        <Out>fileFeatures</Out>
B204:       <Out>restOfResponse</Out></Response>
B205:       <Response Status = "MemoryProblem"><Out>"On Error"</Out>
B206:</Response>
B207:             <Response Status = "OutOfRange"><Out> "OnError" </Out>
B208:</Response>
B209:             <Response Status = "FileOrPatternNotFound"><Out>
"@OnError" B210:</Out> </Response>
B211:             <Response Status = "IncorrectLength"><Out> "OnError"
</Out> B212:</Response>
B213:             <Response Status = "IncorrectP1P2"><Out> "OnError" </Out>
B214:</Response>
B215:             <Response Status = "IncorrectClass"><Out> "OnError" </Out>
B216:</Response>
B217:       </Apdu>
B218:
B219:       <Apdu Id = "ReadBinary">
B220:             <Command>
B221:             <Header Class = "0xA0" Ins = "0xB0" P1 = "offset_1"  P2 =
B222:"offset_2"  Le = "3"/>
B223:             <In>"Void"</In>
B224:             </Command>
B225:             <Response Status = "NormalEnding" Notify = "DoNotNotify">
B226:             <Out>cardBalance </Out></Response>
```

```
B227:              <Response Status = "MemoryProblem"><Out>"On Error"</Out>
B228:</Response>
B229:              <Response Status = "NoneFileSelected"><Out> "OnError"
</Out> B230:</Response>
B231:              <Response Status = "WrongTypeOfFile"><Out> "OnError"
</Out> B232:</Response>
B233:              <Response Status = "WrongAccessCondition" Notify =
"PlugIn">
B234:              <Out>"pluginMessage?secretCode=01"</Out></Response>
B235:              <Response Status = "FileInvalidated"><Out> "OnError"
</Out> B236:</Response>
B237:              <Response Status = "IncorrectP1P2"><Out> "OnError" </Out>
B238:</Response>
B239:              <Response Status = "IncorrectClass"><Out> "OnError" </Out>
B240:</Response>
B241:        </Apdu>
B242:
B243:        <Apdu Id = "Select">
B244:              <Command>
B245:              <Header Class = "0xA0" Ins = "0xA4" P1 = "0"  P2 = "0"
Lc B246:= "2" Le = "0x10"/>
B247:              <In>cardHolderFile</In>
B248:              </Command>
B249:              <Response Status = "NormalEnding" Notify = "DoNotNotify">
B250:              <Out>RFU_1</Out>
B251:              <Out>memoryLeft</Out>
B252:              <Out>fileId</Out>
B253:              <Out>fileType</Out>
B254:              <Out>fileFeatures</Out>
B255:              <Out>restOfResponse</Out></Response>
B256:              <Response Status = "MemoryProblem"><Out>"On Error"</Out>
B257:</Response>
B258:              <Response Status = "OutOfRange"><Out> "OnError" </Out>
B259:</Response>
B260:              <Response Status = "FileOrPatternNotFound"><Out>
"@OnError" B261:</Out> </Response>
B262:              <Response Status = "IncorrectLength"><Out> "OnError"
</Out> B263:</Response>
B264:              <Response Status = "IncorrectP1P2"><Out> "OnError" </Out>
B265:</Response>
B266:              <Response Status = "IncorrectClass"><Out> "OnError" </Out>
B267:</Response>
B268:        </Apdu>
B269:
B270:        <Apdu Id = "ReadBinary">
B271:              <Command>
B272:              <Header Class = "0xA0" Ins = "0xB0" P1 = "offset_1"  P2 =
B273:"offset_2"  Le = "20"/>
B274:              <In>"Void"</In>
B275:              </Command>
B276:              <Response Status = "NormalEnding" Notify = "DoNotNotify">
B277:              <Out>CardHolderName</Out></Response>
B278:              <Response Status = "MemoryProblem"><Out>"On Error"</Out>
B279:</Response>
B280:              <Response Status = "NoneFileSelected"><Out> "OnError"
</Out> B281:</Response>
B282:              <Response Status = "WrongTypeOfFile"><Out> "OnError"
</Out> B283:</Response>
```

```
B284:            <Response Status = "WrongAccessCondition" Notify =
"PlugIn">
B285:            <Out>"pluginMessage?secretCode=01"</Out></Response>
B286:            <Response Status = "FileInvalidated"><Out> "OnError"
</Out> B287:</Response>
B288:            <Response Status = "IncorrectP1P2"><Out> "OnError" </Out>
B289:</Response>
B290:            <Response Status = "IncorrectClass"><Out> "OnError" </Out>
B291:</Response>
B292:       </Apdu>
B293:    </Process>
B294:
B295: <Process Name = "verify">
B296:       <Variable Name = "index" Type = "byte" Default = "0"/>
B297:       <Variable Name = "value" Type = "byteArray" Length = "8"
Default B298:= ""/>
B299:       <Apdu Id = "VerifyCHV">
B300:          <Command>
B301:          <Header Class = "0xA0" Ins = "0x20" P1 = "0x00" P2 =
"index" B302:Lc = "0x08"/>
B303:          <In>value</In>
B304:          </Command>
B305:            <Response Status = "NormalEnding" Notify = "DoNotNotify">
B306:            <Out>"Void"</Out></Response>
B307:             <Response Status = "MemoryProblem"><Out>"On Error"</Out>
B308:</Response>
B309:            <Response Status = "NoneFileSelected"><Out> "OnError"
</Out> B310:</Response>
B311:            <Response Status = "OutOfRange"><Out> "OnError" </Out>
B312:</Response>
B313:            <Response Status = "WrongTypeOfFile"><Out> "OnError"
</Out> B314:</Response>
B315:            <Response Status = "WrongAccessCondition"><Out> "OnError"
B316:</Out> </Response>
B317:            <Response Status = "CHVBlocked"><Out> "OnError" </Out>
B318:</Response>
B319:            <Response Status = "IncorrectLength"><Out> "OnError"
</Out> B320:</Response>
B321:            <Response Status = "IncorrectP1P2"><Out> "OnError" </Out>
B322:</Response>
B323:            <Response Status = "IncorrectClass"><Out> "OnError" </Out>
B324:</Response>
B325:       </Apdu>
B326: </Process>
B327:
B328:    <Process Name = "TransactionAward">
B329:    <Doc>The process represents the ward transaction</Doc>
B330:    <Doc>The members returned are not listed</Doc>
B331:
B332:       <Variable Name = "cardBalanceFile" Type = "byteArray" Default =
B333:"0x2F 0x31"/>
B334:       <Variable Name = "RFU_1" Type = "byteArray" Default = "0x00
B335:0x00"/>
B336:       <Variable Name = "memoryLeft" Type = "byteArray" Default =
"0x00 B337:0x00"/>
B338:       <Variable Name = "fileId" Type = "byteArray" Default = "0x00
B339:0x00"/>
B340:       <Variable Name = "fileType" Type = "byte" Default = "0x00"/>
```

```
B341:       <Variable Name = "fileFeatures" Type = "byteArray" Default =
B342:"0x00 0x00 0x00 0x00 0x00"/>
B343:       <Variable Name = "restOfResponse" Type = "byte" Default =
B344:"0x00"/>
B345:
B346:       <Variable Name = "offset_1" Type = "byte" Default = "0x00"/>
B347:       <Variable Name = "offset_2" Type = "byte" Default = "0x00"/>
B348:        <Variable Name = "cardBalance" Type = "byteArray" Default =
B349:"0x00 0x00 0x00"/>
B350:        <Variable Name = "newBalance" Type = "byteArray" Default =
"0x00 B360:0x00 0x00"/>
B361:
B362:       <Variable Name = "cardInformationFile" Type = "byteArray"
Default B363:= "0x2F 0x30"/>
B364:
B365:        <Variable Name = "date" Type = "byteArray" Default = "0x00
0x00 B366:0x00"/>
B367:     <Variable Name = "amount" Type = "byteArray" Default = "0x00 0x00
B368:0x00"/>
B369:        <Variable Name = "transactionType" Type = "byte" Default =
B370:"0x01"/>
B371:        <Variable Name = "transaction" Type = "byteArray" Length =
"16" B372:Default = ""/>
B373:
B374:       <Variable Name = "cardBalanceFile" Type = "byteArray" Default =
B375:"0x2F 0x31"/>
B376:        <Variable Name = "cardBalance" Type = "byteArray" Default =
B377:"0x00 0x00 0x00"/>
B378:
B379:       <Apdu Id = "Select">
B380:             <Command>
B381:             <Header Class = "0xA0" Ins = "0xA4" P1 = "0"  P2 = "0"
Lc B382:= "2" Le = "0x10"/>
B383:             <In>cardBalanceFile</In>
B384:             </Command>
B385:             <Response Status = "NormalEnding" Notify = "DoNotNotify">
B386:            <Out>RFU_1</Out>
B387:            <Out>memoryLeft</Out>
B388:            <Out>fileId</Out>
B389:            <Out>fileType</Out>
B390:            <Out>fileFeatures</Out>
B391:            <Out>restOfResponse</Out></Response>
B392:             <Response Status = "MemoryProblem"><Out>"On Error"</Out>
B393:</Response>
B394:            <Response Status = "OutOfRange"><Out> "OnError" </Out>
B395:</Response>
B396:            <Response Status = "FileOrPatternNotFound"><Out>
"@OnError" B397:</Out> </Response>
B398:            <Response Status = "IncorrectLength"><Out> "OnError"
</Out> B399:</Response>
B400:            <Response Status = "IncorrectP1P2"><Out> "OnError" </Out>
B401:</Response>
B402:            <Response Status = "IncorrectClass"><Out> "OnError" </Out>
B403:</Response>
B404:       </Apdu>
B405:
B406:       <Apdu Id = "ReadBinary">
B407:             <Command>
```

```
B408:              <Header Class = "0xA0" Ins = "0xB0" P1 = "offset_1"  P2 =
B409:"offset_2"  Le = "3"/>
B410:              <In>"Void"</In>
B411:               </Command>
B412:               <Response Status = "NormalEnding" Notify = "DoNotNotify">
B413:              <Out>cardBalance</Out></Response>
B414:               <Response Status = "MemoryProblem"><Out>"On Error"</Out>
B415:</Response>
B416:              <Response Status = "NoneFileSelected"><Out> "OnError"
</Out> B417:</Response>
B418:              <Response Status = "WrongTypeOfFile"><Out> "OnError"
</Out> B419:</Response>
B420:              <Response Status =
B421:"WrongAccessCondition"><Out>"OnError"</Out></Response>
B422:              <Response Status = "FileInvalidated"><Out> "OnError"
</Out> B423:</Response>
B424:              <Response Status = "IncorrectP1P2"><Out> "OnError" </Out>
B425:</Response>
B426:              <Response Status = "IncorrectClass"><Out> "OnError" </Out>
B427:</Response>
B428:        </Apdu>
B429:
B430:         <Apdu Id = "Callback">
B431:            <Command>
B432:            <Header Class = "0x00" Ins = "0x00" P1 = "0x00"  P2 =
"0x00" B432: Lc = "0x00" Le = "0x00"/>
B434:            <In>"awardBalance?balance=cardBalance"</In>
B435:            </Command>
B436:            <Response Status = "NormalEnding" Notify = "DoNotNotify">
B437:             <Out>newBalance</Out></Response>
B438:      </Apdu>
B439:
B440:        <Apdu Id = "Callback">
B441:           <Command>
B442:           <Header Class = "0x00" Ins = "0x00" P1 = "0x00"  P2 = "0x00"
 B443:Lc = "0x00" Le = "0x00"/>
B444:           <In>"crypto?award=DES"</In>
B445:           </Command>
B446:           <Response Status = "NormalEnding" Notify =
B447:"DoNotNotify"><Out>"Void"</Out></Response>
B448:      </Apdu>
B449:
B450:        <Apdu Id = "Select">
B451:             <Command>
B452:             <Header Class = "0xA0" Ins = "0xA4" P1 = "0"  P2 = "0"
Lc B453:= "2" Le = "0x10"/>
B454:             <In>cardInformationFile</In>
B455:             </Command>
B456:             <Response Status = "NormalEnding" Notify = "DoNotNotify">
B457:            <Out>RFU_1</Out>
B458:            <Out>memoryLeft</Out>
B459:            <Out>fileId</Out>
B460:            <Out>fileType</Out>
B461:            <Out>fileFeatures</Out>
B462:            <Out>restOfResponse</Out></Response>
B463:             <Response Status = "MemoryProblem"><Out>"On Error"</Out>
B464:</Response>
```

```
B465:            <Response Status = "OutOfRange"><Out> "OnError" </Out>
B466:</Response>
B467:            <Response Status = "FileOrPatternNotFound"><Out>
"@OnError" B468:</Out> </Response>
B469:            <Response Status = "IncorrectLength"><Out> "OnError"
</Out> B470:</Response>
B471:            <Response Status = "IncorrectP1P2"><Out> "OnError" </Out>
B472:</Response>
B473:            <Response Status = "IncorrectClass"><Out> "OnError" </Out>
B474:</Response>
B475:       </Apdu>
B476:
B477:       <Apdu Id = "UpdateBinary">
B478:            <Command>
B479:            <Header Class = "0xA0" Ins = "0xD6" P1 = "offset_1"  P2 =
B480:"offset_2"  Lc = "0x17"/>
B481:            <In>date</In>
B482:           <In>amount</In>
B483:            <In>transactionType</In>
B484:            <In>transaction</In>
B485:            </Command>
B486:            <Response Status = "NormalEnding" Notify = "DoNotNotify">
B487:            <Out>"Void"</Out></Response>
B488:            <Response Status = "MemoryProblem"><Out>"On Error"</Out>
B489:</Response>
B490:            <Response Status = "NoneFileSelected"><Out> "OnError"
</Out> B491:</Response>
B492:            <Response Status = "WrongTypeOfFile"><Out> "OnError"
</Out> B493:</Response>
B494:            <Response Status = "WrongAccessCondition"><Out> "OnError"
B495:</Out> </Response>
B496:            <Response Status = "FileInvalidated"><Out> "OnError"
</Out> B497:</Response>
B498:            <Response Status = "IncorrectP1P2"><Out> "OnError" </Out>
B499:</Response>
B500:            <Response Status = "IncorrectClass"><Out> "OnError" </Out>
B501:</Response>
B502:       </Apdu>
B503:
B504:       <Apdu Id = "Select">
B505:            <Command>
B506:            <Header Class = "0xA0" Ins = "0xA4" P1 = "0"  P2 = "0"
Lc B507:= "2" Le = "0x10"/>
B508:            <In>cardBalanceFile</In>
B509:            </Command>
B510:            <Response Status = "NormalEnding" Notify = "DoNotNotify">
B511:            <Out>RFU_1</Out>
B512:            <Out>memoryLeft</Out>
B513:            <Out>fileId</Out>
B514:            <Out>fileType</Out>
B515:            <Out>fileFeatures</Out>
B516:            <Out>restOfResponse</Out></Response>
B517:            <Response Status = "MemoryProblem"><Out>"On Error"</Out>
B518:</Response>
B519:            <Response Status = "OutOfRange"><Out> "OnError" </Out>
B520:</Response>
B521:            <Response Status = "FileOrPatternNotFound"><Out>
"@OnError" B522:</Out> </Response>
```

```
B523:            <Response Status = "IncorrectLength"><Out> "OnError"
</Out> B524:</Response>
B525:            <Response Status = "IncorrectP1P2"><Out> "OnError" </Out>
B526:</Response>
B527:            <Response Status = "IncorrectClass"><Out> "OnError" </Out>
B528:</Response>
B529:       </Apdu>
B530:
B531:       <Apdu Id = "UpdateBinary">
B532:        <Command>
B533:        <Header Class = "0xA0" Ins = "0xD6" P1 = "offset_1"  P2 =
B534:"offset_2"  Lc = "3"/>
B535:            <In>newBalance</In>
B536:        </Command>
B537:        <Response Status = "NormalEnding" Notify = "DoNotNotify">
B538:            <Out>"Void"</Out></Response>
B539:        <Response Status = "MemoryProblem"><Out>"On Error"</Out>
B540:</Response>
B541:            <Response Status = "NoneFileSelected"><Out> "OnError"
</Out> B542:</Response>
B543:           <Response Status = "WrongTypeOfFile"><Out> "OnError" </Out>
B544:</Response>
B545:            <Response Status = "WrongAccessCondition"><Out> "OnError"
B546:</Out> </Response>
B547:            <Response Status = "FileInvalidated"><Out> "OnError"
</Out> B548:</Response>
B549:            <Response Status = "IncorrectP1P2"><Out> "OnError" </Out>
B550:</Response>
B551:            <Response Status = "IncorrectClass"><Out> "OnError" </Out>
B552:</Response>
B553:       </Apdu>
B554:    </Process>
B555:    <Process Name = "TransactionRedeem">
B556:    <Doc>The process represents the redeem transaction </Doc>
B557:    <Doc>The members returned are not listed</Doc>
B558:
B559:       <Variable Name = "cardBalanceFile" Type = "byteArray" Default =
B560:"0x2F 0x31"/>
B561:       <Variable Name = "RFU_1" Type = "byteArray" Default = "0x00
B562:0x00"/>
B563:       <Variable Name = "memoryLeft" Type = "byteArray" Default =
"0x00 B564:0x00"/>
B565:       <Variable Name = "fileId" Type = "byteArray" Default = "0x00
B566:0x00"/>
B567:       <Variable Name = "fileType" Type = "byte" Default = "0x00"/>
B568:       <Variable Name = "fileFeatures" Type = "byteArray" Default =
B569:"0x00 0x00 0x00 0x00 0x00"/>
B570:       <Variable Name = "restOfResponse" Type = "byte" Default =
B571:"0x00"/>
B572:
B573:       <Variable Name = "offset_1" Type = "byte" Default = "0x00"/>
B574:       <Variable Name = "offset_2" Type = "byte" Default = "0x00"/>
B575:        <Variable Name = "cardBalance" Type = "byteArray" Default =
B576:"0x00 0x00 0x00"/>
B577:        <Variable Name = "newBalance" Type = "byteArray" Default =
"0x00 B578:0x00 0x00"/>
B579:
```

```
B580:       <Variable Name = "cardInformationFile" Type = "byteArray"
Default B581:= "0x2F 0x30"/>
B582:
B583:        <Variable Name = "date" Type = "byteArray" Default = "0x00
0x00 B584:0x00"/>
B585:     <Variable Name = "amount" Type = "byteArray" Default = "0x00 0x00
B586:0x00"/>
B587:        <Variable Name = "transactionType" Type = "byte" Default =
B588:"0x02"/>
B589:        <Variable Name = "transaction" Type = "byteArray" Length =
"16" B590:Default = ""/>
B591:
B592:       <Variable Name = "cardBalanceFile" Type = "byteArray" Default =
B593:"0x2F 0x31"/>
B594:        <Variable Name = "cardBalance" Type = "byteArray" Default =
B595:"0x00 0x00 0x00"/>
B596:
B597:       <Apdu Id = "Select">
B598:            <Command>
B599:            <Header Class = "0xA0" Ins = "0xA4" P1 = "0"  P2 = "0"
Lc B600:= "2" Le = "0x10"/>
B601:            <In>cardBalanceFile</In>
B602:            </Command>
B603:            <Response Status = "NormalEnding" Notify = "DoNotNotify">
B604:           <Out>RFU_1</Out>
B605:           <Out>memoryLeft</Out>
B606:           <Out>fileId</Out>
B607:           <Out>fileType</Out>
B608:           <Out>fileFeatures</Out>
B609:           <Out>restOfResponse</Out></Response>
B610:            <Response Status = "MemoryProblem"><Out>"On Error"</Out>
B611:</Response>
B612:           <Response Status = "OutOfRange"><Out> "OnError" </Out>
B613:</Response>
B614:           <Response Status = "FileOrPatternNotFound"><Out>
"@OnError" B615:</Out> </Response>
B616:           <Response Status = "IncorrectLength"><Out> "OnError"
</Out> B617:</Response>
B618:           <Response Status = "IncorrectP1P2"><Out> "OnError" </Out>
B619:</Response>
B620:           <Response Status = "IncorrectClass"><Out> "OnError" </Out>
B621:</Response>
B622:      </Apdu>
B623:
B624:       <Apdu Id = "ReadBinary">
B625:            <Command>
B626:            <Header Class = "0xA0" Ins = "0xB0" P1 = "offset_1"  P2 =
B627:"offset_2"  Le = "3"/>
B628:           <In>"Void"</In>
B629:            </Command>
B630:            <Response Status = "NormalEnding" Notify = "DoNotNotify">
B631:           <Out>cardBalance</Out></Response>
B632:            <Response Status = "MemoryProblem"><Out>"On Error"</Out>
B633:</Response>
B634:           <Response Status = "NoneFileSelected"><Out> "OnError"
</Out> B635:</Response>
B636:           <Response Status = "WrongTypeOfFile"><Out> "OnError"
</Out> B637:</Response>
```

```
B638:            <Response Status =
B639:"WrongAccessCondition"><Out>"OnError"</Out></Response>
B640:            <Response Status = "FileInvalidated"><Out> "OnError"
</Out> B641:</Response>
B642:            <Response Status = "IncorrectP1P2"><Out> "OnError" </Out>
B643:</Response>
B644:            <Response Status = "IncorrectClass"><Out> "OnError" </Out>
B645:</Response>
B646:       </Apdu>
B647:
B648:        <Apdu Id = "Callback">
B649:           <Command>
B650            <Header Class = "0x00" Ins = "0x00" P1 = "0x00"  P2 =
"0x00" B651: Lc = "0x00" Le = "0x00"/>
B652:           <In>"redeemBalance?balance=cardBalance"</In>
B653:           </Command>
B654:           <Response Status = "NormalEnding" Notify = "DoNotNotify">
B655:            <Out>newBalance </Out></Response>
B656:     </Apdu>
B657:
B658:       <Apdu Id = "Callback">
B659:          <Command>
B660:          <Header Class = "0x00" Ins = "0x00" P1 = "0x00"  P2 = "0x00"
 B661:Lc = "0x00" Le = "0x00"/>
B662:          <In>"crypto?award=DES"</In>
B663:          </Command>
B664:          <Response Status = "NormalEnding" Notify =
B665:"DoNotNotify"><Out>"Void"</Out></Response>
B666:     </Apdu>
B667:
B668:       <Apdu Id = "Select">
B669:            <Command>
B670:            <Header Class = "0xA0" Ins = "0xA4" P1 = "0"  P2 = "0"
Lc B671: = "2" Le = "0x10"/>
B672:            <In>cardInformationFile</In>
B673:            </Command>
B674:            <Response Status = "NormalEnding" Notify = "DoNotNotify">
B675:           <Out>RFU_1</Out>
B676:           <Out>memoryLeft</Out>
B677:           <Out>fileId</Out>
B678:           <Out>fileType</Out>
B679:           <Out>fileFeatures</Out>
B680:           <Out>restOfResponse</Out></Response>
B691:            <Response Status = "MemoryProblem"><Out>"On Error"</Out>
B692:</Response>
B693:            <Response Status = "OutOfRange"><Out> "OnError" </Out>
B694:</Response>
B695:            <Response Status = "FileOrPatternNotFound"><Out>
"@OnError" B696:</Out> </Response>
B697:            <Response Status = "IncorrectLength"><Out> "OnError"
</Out> B698:</Response>
B699:            <Response Status = "IncorrectP1P2"><Out> "OnError" </Out>
B700:</Response>
B701:            <Response Status = "IncorrectClass"><Out> "OnError" </Out>
B702:</Response>
B703:       </Apdu>
B704:
B705:       <Apdu Id = "UpdateBinary">
```

```
B706:              <Command>
B707:              <Header Class = "0xA0" Ins = "0xD6" P1 = "offset_1"  P2 =
B708:"offset_2"  Lc = "0x17"/>
B709:              <In>date</In>
B710:             <In>amount</In>
B711:              <In>transactionType</In>
B712:              <In>transaction</In>
B713:              </Command>
B714:              <Response Status = "NormalEnding" Notify = "DoNotNotify">
B715:              <Out>"Void"</Out></Response>
B716:              <Response Status = "MemoryProblem"><Out>"On Error"</Out>
B717:</Response>
B718:              <Response Status = "NoneFileSelected"><Out> "OnError"
</Out> B719:</Response>
B720:              <Response Status = "WrongTypeOfFile"><Out> "OnError"
</Out> B721:</Response>
B722:              <Response Status = "WrongAccessCondition"><Out> "OnError"
B723:</Out> </Response>
B724:              <Response Status = "FileInvalidated"><Out> "OnError"
</Out> B725:</Response>
B726:              <Response Status = "IncorrectP1P2"><Out> "OnError" </Out>
B727:</Response>
B728:              <Response Status = "IncorrectClass"><Out> "OnError" </Out>
B729:</Response>
B730:        </Apdu>
B731:
B732:        <Apdu Id = "Select">
B733:              <Command>
B734:              <Header Class = "0xA0" Ins = "0xA4" P1 = "0"  P2 = "0"
Lc B735:= "2" Le = "0x10"/>
B736:              <In>cardBalanceFile</In>
B737:              </Command>
B739:              <Response Status = "NormalEnding" Notify = "DoNotNotify">
B740:             <Out>RFU_1</Out>
B741:             <Out>memoryLeft</Out>
B742:             <Out>fileId</Out>
B743:             <Out>fileType</Out>
B744:             <Out>fileFeatures</Out>
B745:             <Out>restOfResponse</Out></Response>
B746:              <Response Status = "MemoryProblem"><Out>"On Error"</Out>
B747:</Response>
B748:             <Response Status = "OutOfRange"><Out> "OnError" </Out>
B749:</Response>
B750:             <Response Status = "FileOrPatternNotFound"><Out>
"@OnError" B751:</Out> </Response>
B752:             <Response Status = "IncorrectLength"><Out> "OnError"
</Out> B753:</Response>
B754:             <Response Status = "IncorrectP1P2"><Out> "OnError" </Out>
B755:</Response>
B756:             <Response Status = "IncorrectClass"><Out> "OnError" </Out>
B757:</Response>
B758:        </Apdu>
B759:
B760:        <Apdu Id = "UpdateBinary">
B761:              <Command>
B762:              <Header Class = "0xA0" Ins = "0xD6" P1 = "offset_1"  P2 =
B763:"offset_2"  Lc = "3"/>
B764:             <In>newBalance</In>
```

```
B765:              </Command>
B766:              <Response Status = "NormalEnding" Notify = "DoNotNotify">
N767:             <Out>"Void"</Out></Response>
B769:              <Response Status = "MemoryProblem"><Out>"On Error"</Out>
B770:</Response>
B771:             <Response Status = "NoneFileSelected"><Out> "OnError"
</Out> B772:</Response>
B773:             <Response Status = "WrongTypeOfFile"><Out> "OnError"
</Out> B774:</Response>
B775:             <Response Status = "WrongAccessCondition"><Out> "OnError"
B776:</Out> </Response>
B777:             <Response Status = "FileInvalidated"><Out> "OnError"
</Out> B778:</Response>
B779:             <Response Status = "IncorrectP1P2"><Out> "OnError" </Out>
B780:</Response>
B781:             <Response Status = "IncorrectClass"><Out> "OnError" </Out>
B782:</Response>
B783:       </Apdu>
B784:    </Process>
B785:</Profile>
B786:
B787:<StatusWord>
B788: <SW Verbose = "NormalEnding" Status = "0x9000"/>
B789: <SW Verbose = "MemoryProblem" Status = "0x9240"/>
B790: <SW Verbose = "NoneFileSelected" Status = "0x9400"/>
B791: <SW Verbose = "OutOfRange" Status = "0x9402"/>
B792: <SW Verbose = "FileOrPatternNotFound" Status = "0x9404"/>
B793: <SW Verbose = "WrongTypeOfFile" Status = "0x9408"/>
B794: <SW Verbose = "CHVDoesNotExist" Status = "0x9802"/>
B795: <SW Verbose = "WrongAccessCondition" Status = "0x9804"/>
B796: <SW Verbose = "InContradictionWithTheCHVStatus" Status = "0x9808"/>
B797: <SW Verbose = "CHVBlocked" Status = "0x9840"/>
B798: <SW Verbose = "FileInvalidated" Status = "0x9810"/>
B799: <SW Verbose = "IncorrectLength" Status = "0x6700"/>
B800: <SW Verbose = "IncorrectP1P2" Status = "0x6B00"/>
B801:    <SW Verbose = "IncorrectClass" Status = "0x6E00"/>
B802:</StatusWord>
B803:</CardDocument>
```

APPLICATION BOOT

The example of a boot file listed below from line C1 through line C177 provides the data and rules for testing a dictionary for compatibility with the example application logic unit listed above. The rules described in the boot file for compatibility testing allow two types of smart cards to be approved as compatible with the example application logic unit. Specifically, the example boot file allows Gemplus GemXplore98, and Gemplus GemClub™ smart cards to be scanned and approved for compatibility with the application logic unit. The procedure implemented according to the rules of the boot file yields a unique,

predetermined value for each of these three compatible smart cards. This unique predetermined value is returned to the linking engine as the output value of the boot process and is used to test the compatibility of the card, as described in connection with **Figure 6**. Additionally, the boot file includes an address where the dictionaries can be retrieved for compatibility testing.

```
C1:<?xml version = "1.0" ?>
C2:<!DOCTYPE CardDocument SYSTEM "..\dtd\com\gemplus\sml.dtd">
C3:
C4:<!-- * Copyright (c) 1997 - 1999 Gemplus group. All Rights Reserved.
 C5:     -->
C6:
C7:<CardDocument>
C8:<Profile Type = "GemXplore98" Version = "0.1">
C9:   <Signature Type = "Application">"signatures"</Signature>
C10:
C11:  <Process Name = "signatures">
C12:      <Doc>This is the description of the getFile process to retreive
the C13:path</Doc>
C14:      <Doc> of the XML file related to the Gemplus' GemXpresso smart
C15:card</Doc>
C16:    <Doc> step 1 - Select CardSerialNumber file </Doc>
C17:    <Doc> step 2 - Select GSM directory </Doc>
C18:    <Doc> step 3 - Select IMSI file </Doc>
C19:    <Doc> step 4 - Select Telecom directory </Doc>
C20:
C21:      <Variable Name = "cardSerialNumberFile" Type = "byteArray"
Default C22:= "0x2F 0xE2"/>
C23:      <Variable Name = "gsmDirectory" Type = "byteArray" Default =
"0x7F C24:0x20"/>
C25:      <Variable Name = "imsiFile" Type = "byteArray" Default = "0x6F
C26:0x09"/>
C27:      <Variable Name = "telecomDirectory" Type = "byteArray" Default =
C28:"0x7F 0x10"/>
C29:      <Variable Name = "root" Type = "byteArray" Default = "0x3F
0x00"/>
C30:    <Variable Name = "RFU_1" Type = "byteArray" Default = "0x00 0x00"/>
C31:    <Variable Name = "fileId" Type = "byteArray" Default = "0x00
0x00"/>
C32:      <Variable Name = "memoryLeft" Type = "byteArray" Default = "0x00
C33:0x00"/>
C34:      <Variable Name = "fileType" Type = "byte" Default = "0x00"/>
C35:      <Variable Name = "fileFeatures" Type = "byteArray" Default =
"0x00 C36:0x00 0x00 0x00 0x00"/>
C37:      <Variable Name = "restOfResponse" Type = "byte" Default =
"0x00"/>
C38:    <Apdu Id = "Select">
C39:        <Command>
C40:          <Header Class = "0xA0" Ins = "0xA4" P1 = "0"  P2 = "0"  Lc =
C41:"2" Le = "0x20"/>
C42:          <In>cardSerialNumberFile</In>
```

```
C43:          </Command>
C44:          <Response Status = "NormalEnding" Notify = "DoNotNotify">
C45:            <Out>RFU_1</Out>
C46:            <Out>memoryLeft</Out>
C47:            <Out>fileId</Out>
C48:            <Out>fileType</Out>
C49:            <Out>fileFeatures</Out>
C50:            <Out>restOfResponse</Out></Response>
C51:      </Apdu>
C52:
C53:      <Apdu Id = "Select">
C54:          <Command>
C55:            <Header Class = "0xA0" Ins = "0xA4" P1 = "0"  P2 = "0"  Lc =
C56:"2" Le = "0x20"/>
C57:            <In>gsmDirectory</In>
C58:          </Command>
C59:          <Response Status = "NormalEnding" Notify = "DoNotNotify">
C60:            <Out>RFU_1</Out>
C61:            <Out>memoryLeft</Out>
C62:            <Out>fileId</Out>
C63:            <Out>fileType</Out>
C64:            <Out>fileFeatures</Out>
C65:            <Out>restOfResponse</Out></Response>
C66:      </Apdu>
C67:
C68:      <Apdu Id = "Select">
C69:          <Command>
C70:            <Header Class = "0xA0" Ins = "0xA4" P1 = "0"  P2 = "0"  Lc =
C71:"2" Le = "0x20"/>
C72:            <In>imsiFile</In>
C73:          </Command>
C74:          <Response Status = "NormalEnding" Notify = "DoNotNotify">
C75:            <Out>RFU_1</Out>
C76:            <Out>memoryLeft</Out>
C77:            <Out>fileId</Out>
C78:            <Out>fileType</Out>
C79:            <Out>fileFeatures</Out>
C80:            <Out>restOfResponse</Out></Response>
C81:      </Apdu>
C82:
C83:      <Apdu Id = "Select">
C84:          <Command>
C85:            <Header Class = "0xA0" Ins = "0xA4" P1 = "0"  P2 = "0"  Lc =
C86:"2" Le = "0x20"/>
C87:            <In>telecomDirectory</In>
C88:          </Command>
C89:          <Response Status = "NormalEnding" Notify = "DoNotNotify">
C90:            <Out>RFU_1</Out>
C91:            <Out>memoryLeft</Out>
C92:            <Out>fileId</Out>
C93:            <Out>fileType</Out>
C94:            <Out>fileFeatures</Out>
C95:            <Out>restOfResponse</Out></Response>
C96:      </Apdu>
C97:  </Process>
C98:
C99:  <Process Name = "getFile">
```

```
C100:    <Doc>This is the description of the getFile process to retreive the
C101:path</Doc>
C102:    <Doc> of the XML file related to the Gemplus' GemXpresso smart
C103:card</Doc>
C104:    <Apdu Id = "returnFilename">
C105:        <Command>
C106:          <Header Class = "0x00" Ins = "0x00" P1 = "0"  P2 = "0"  Le =
C107:"0x00"/>
C108:          <In>"file:////d:\\my
C109:developments\\xml\\sml\Samples\GsmLoyaltyA.sml"</In>
C110:      </Command>
C111:       <Response Status = "NormalEnding" Notify = "DoNotNotify">
C112:          <Out>"Void"</Out></Response>
C113:    </Apdu>
C114: </Process>
C115:</Profile>
C116:
C117:<Profile Type = "GemClub" Version = "0.1">
C118:    <Signature Type = "Application">"signature"</Signature>
C119:
C120: <Process Name = "signature">
C121:    <Doc>This is the description of the getFile process to retreive the
C122:path</Doc>
C123:    <Doc> of the XML file related to the Gemplus' GemClub smart
C124:card</Doc>
C125:    <Doc> step 1 - get the card information</Doc>
C126:
C127:       <Variable Name = "CardSerialNumber" Type = "byteArray" Length =
C128:"8" Default = ""/>
C129:       <Variable Name = "Unknown1" Type = "byteArray" Length = "8"
C130:Default = ""/>
C131:       <Variable Name = "Unknown2" Type = "byteArray" Length = "8"
C132:Default = ""/>
C133:       <Variable Name = "IssuerReference" Type = "byteArray" Length =
"4" C134:Default = ""/>
C135:    <Apdu Id = "GetCardInfo">
C136:        <Command>
C137:          <Header Class = "0x80" Ins = "0xBE" P1 = "0" P2 = "0" Le =
C138:"0x1C"/>
C139:          <In>"Void"</In>
C140:        </Command>
C141:        <Response Status = "NormalEnding" Notify = "DoNotNotify">
C142:          <Out>Unknown1</Out>
C143:          <Out>Unknown2</Out>
C144:          <Out>CardSerialNumber</Out>
C145:          <Out>IssuerReference</Out></Response>
C146:    </Apdu>
C147:     </Process>
C148:
C149: <Process Name = "getFile">
C150:         <Doc>This is the description of the getFile process to
retreive C151:the path</Doc>
C152:         <Doc> of the XML file related to the Gemplus' GemClub smart
C153:card</Doc>
C154:      <Apdu Id = "returnFilename">
C155:            <Command>
C156:            <Header Class = "0x00" Ins = "0x00" P1 = "0"  P2 = "0"  Lc
= C157:"0" Le = "0"/>
```

```
C158:            <In>"file:////c:\\my
C159:developments\\xml\\sml\Samples\GemClubLoyalty.sml"</In>
C160:            </Command>
C161:          <Response Status = "NormalEnding" Notify = "DoNotNotify">
C162:            <Out>"Void"</Out></Response>
C163:        </Apdu>
C164:    </Process>
C165:</Profile>
C166:
C167:<StatusWord>
C168: <SW Verbose = "NormalEnding" Status = "0x9000"/>
C169:</StatusWord>
C170:
C171:<Manifest>
C172: <Principal/>
C173: <Digest></Digest>
C174: <DicSignature></DicSignature>
C175: <Certificate></Certificate>
C176:</Manifest>
C177:</CardDocument>
```

While the application protocol and boot file examples described above are written in XML, alternate embodiments of the present invention could use different versions of XML, or different languages or implementations, to provide the description. Additionally, the boot file and application protocol need not be written in the same language, as is done in the present example.

While the application logic unit in the above example is written in Java, alternate embodiments of the present invention could use other computer languages for the implementation of the logic of the application.

The present invention allows a developer of application software to write an application logic unit without concern for the particular hardware implementation of readers, computers, or smart cards. Through the linking engine, the present invention retrieves the hardware specific description of the logical processes of the application logic unit. The linking engine matches the identifiers specifying the hardware used, the reader, smart card and any computer, to the process called for the in application logic unit. Accordingly, the present invention has the advantage of allowing an application developer to write applications that may be implemented on any combination of terminal or smart device with an existing dictionary for the combination of terminal and smart device. The present invention reduces the burden on the memory of the terminal by allowing hardware specific portions of the

application to be downloaded for a specific smart device, thereby eliminating the need to store hardware specific functional elements of the application related to the smart device.

The present invention allows other smart cards, or other types of smart devices, to be added as compatible with the application logic unit by either adding to the existing boot file, or using a new boot file with the scanning information for the new smart card. In this manner the present invention, as illustrated by the examples contained herein, provides enhanced flexibility and interoperability among different smart device types through the addition of boot files and dictionaries without the need to modify the existing application logic unit.

While the preferred embodiment of the present invention links the application protocol to the application logic unit, alternate embodiments of the present invention may perform the linking operation prior to run time.

CLAIMS

I claim:

1. A terminal for running an application program for processing data in connection with a smart device, comprising:

an application logic unit providing the hardware independent description of the logic of the application program; and

an identifier specifying the location of an application protocol providing the hardware description of the application program.

2. The terminal of claim 1, wherein the terminal further comprises:

a means for linking the application logic unit to the application protocol to implement the logical processes of the application logic unit in connection with the smart device.

3. The terminal of claim 1, wherein the identifier corresponds to a memory location on the smart device.

4. The terminal of claim 1, wherein the identifier corresponds to a memory location accessible through a network connection.

5. The terminal of claim 1, wherein the identifier corresponds to a memory location on the terminal.

6. The terminal of claim 2, wherein the means for linking performs a compatibility test prior to linking the application protocol to the application logic unit.

7. The terminal of claim 6, wherein the means for linking performs the compatibility test on a plurality of dictionaries prior to selecting a dictionary for sue as the application protocol.

8. A smart device used in connection with a terminal for running an application program for processing data, comprising:

a memory unit; and

an identifier located in the memory unit specifying the location of an application protocol, the application protocol providing the hardware description of the application program.

9. A method for running an application program for processing data in connection with a smart device, the method comprising the steps of:

initiating the running of an application logic unit;

initiating the testing of at least one dictionary for compatibility with the hardware implementation,

selecting a dictionary for use as the application protocol based on the outcome of the compatibility test;

linking the selected dictionary to the application logic unit for use as the application protocol; and

implementing the logic of the application logic unit using the definitions contained in the application protocol.

10. A method of linking a dictionary to an application logic unit to implement the application logic unit, comprising:

receiving a request from an application logic unit to find a dictionary compatible with the hardware implementation present at the time the application logic unit makes the request;

retrieving a list including at least one identifier of a potentially compatible dictionary;

selecting a dictionary for compatibility testing from the list of potentially compatible dictionaries;

retrieving the selected dictionary;

running a boot to scan the dictionary for segments of the dictionary used in the compatibility test;

generating a output value corresponding to the selected dictionary;

comparing the output value to a bootcheck value;

including the selected dictionary on a compatible dictionary list if the output value is equivalent to the bootcheck value;

selecting a dictionary from the compatible dictionary list for use as the application protocol; and

linking the dictionary selected from the compatible dictionary list to the application logic unit for use in implementing the application logic unit.

11. The method of linking the application logic unit to the application protocol of claim 10, wherein at least one additional dictionary is selected for compatibility testing.

12. The method of linking the application logic unit to the application protocol of claim 11, wherein additional dictionaries are included on the compatible dictionary list when the corresponding output value of the tested dictionary is equivalent to the bootcheck value.

13. The method of linking the application logic unit to the application protocol of claim 11 wherein all the dictionaries on the list of potentially compatible dictionaries are tested for compatibility.

14. The method of linking the application logic unit to the application protocol of claim 11 wherein all dictionaries are included on the compatible dictionary list where the dictionary output value is equivalent to the bootcheck value.

15. The method of linking the application logic unit to the application protocol of claim 13 wherein all dictionaries are included on the compatible dictionary list where the dictionary output value is equivalent to the bootcheck value.

16. A method of running an application program, comprising the steps of:

creating a profile for the application protocol;

establishing a linking engine;

initiating a test of at least one dictionary for compatibility with the hardware implementation and an application logic unit;

selecting a dictionary for use of as the application protocol based on the outcome of the compatibility test;

invoking at least one smart device method; and

receiving an output response based on a definition of the invoked smart device method, the definition of the invoked smart card method contained in the application protocol.

17. A application program for processing data in connection with data processed on a smart device, comprising:

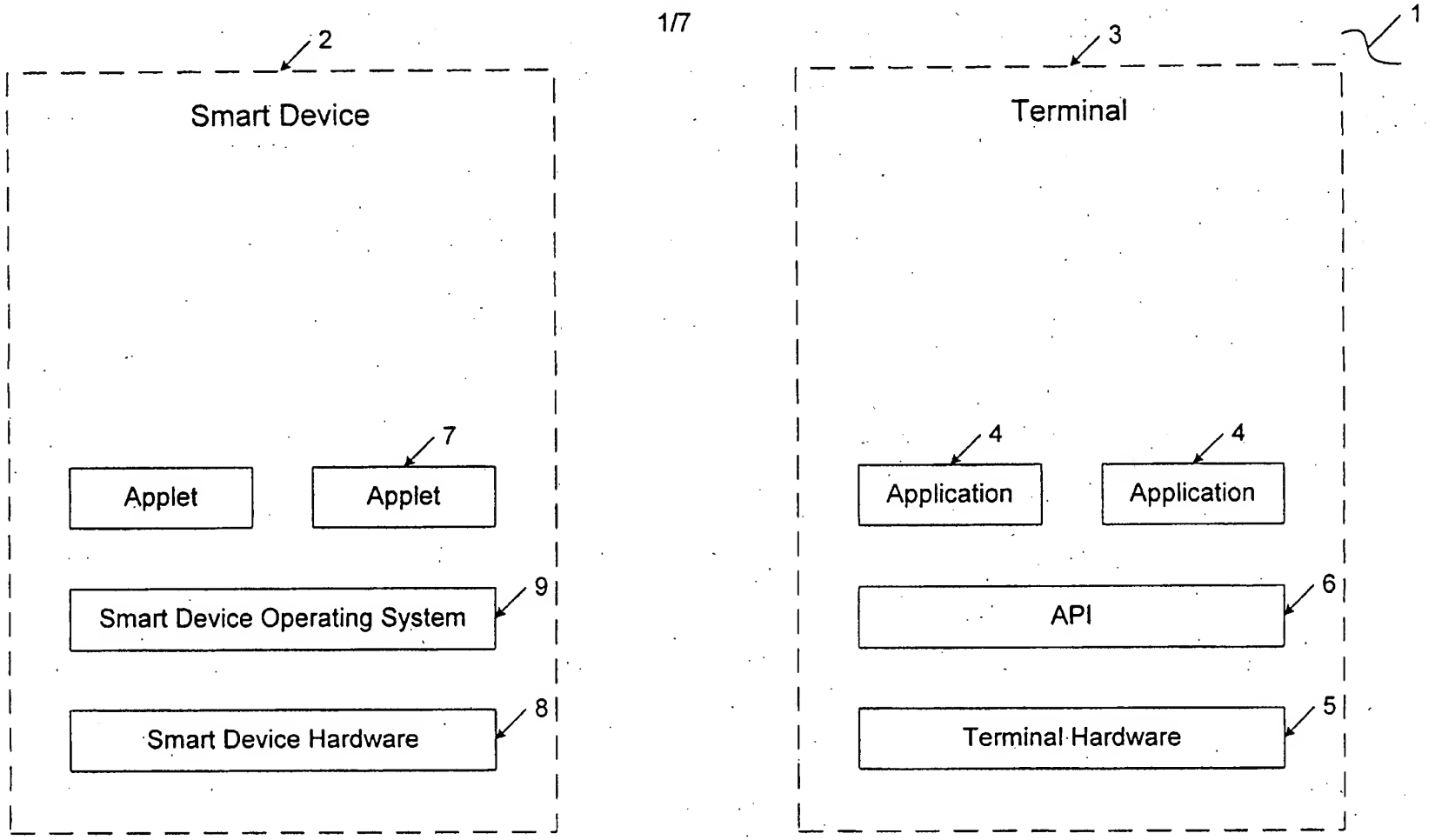
an application logic unit specifying the logic performed in processing the data, the application logic unit including at least one smart device method; and

an application protocol including a definition of the smart device method, the definition providing the hardware specific component of the smart device method.

18. The application program of claim 17, further comprising a boot file, the boot file specifying segments of a dictionary tested for compatibility as the application protocol for the application logic unit.

19. The application program of claim 17, wherein the application protocol is linked to the application logic unit by a linking engine.

20. The application program of claim 19, wherein the linking engine performs a compatibility test prior to linking the application protocol to the application logic unit.

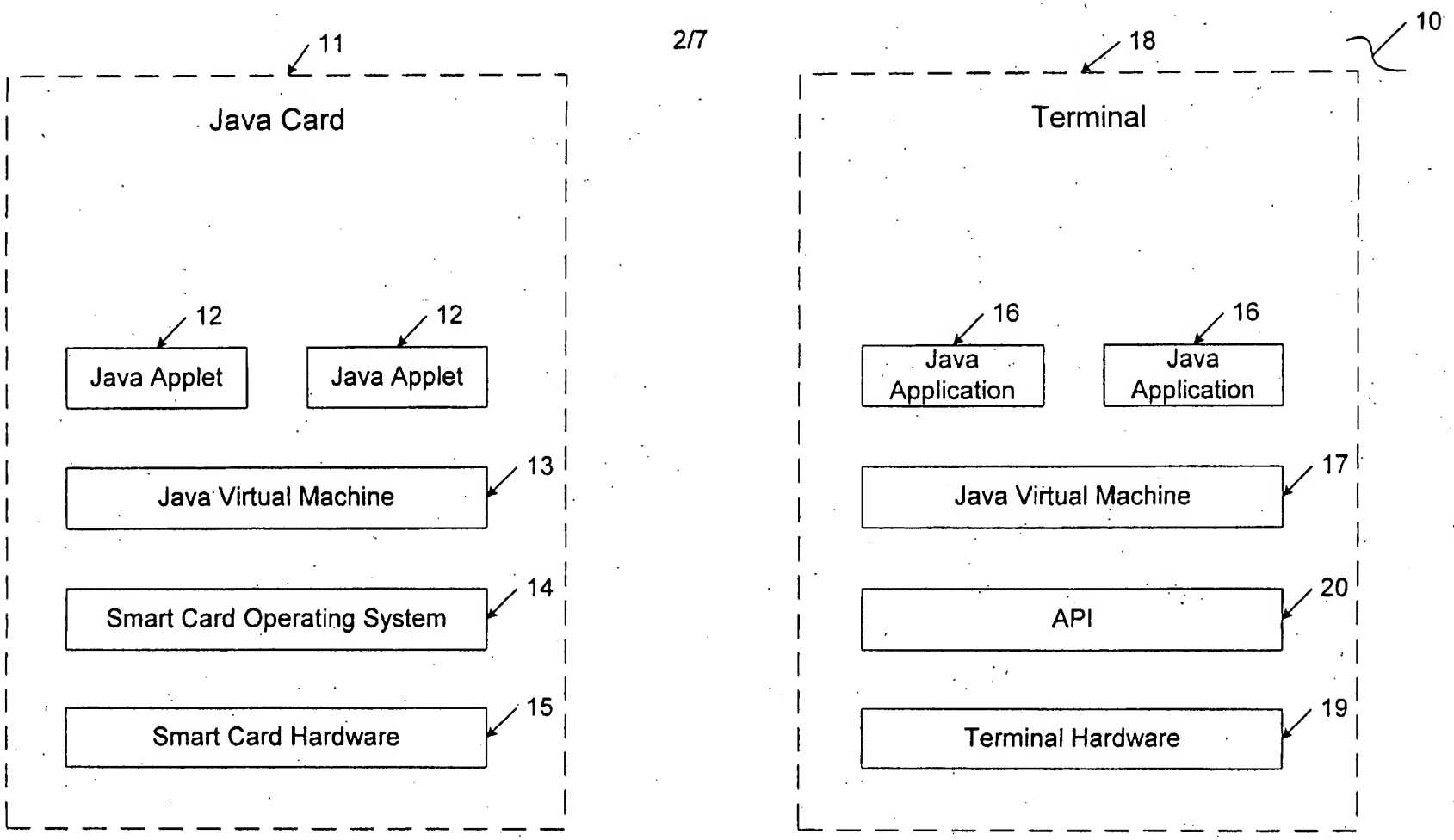


Prior Art

Figure 1

WO 00/75775 PCT/US00/15689

1/7



Prior Art

Figure 2

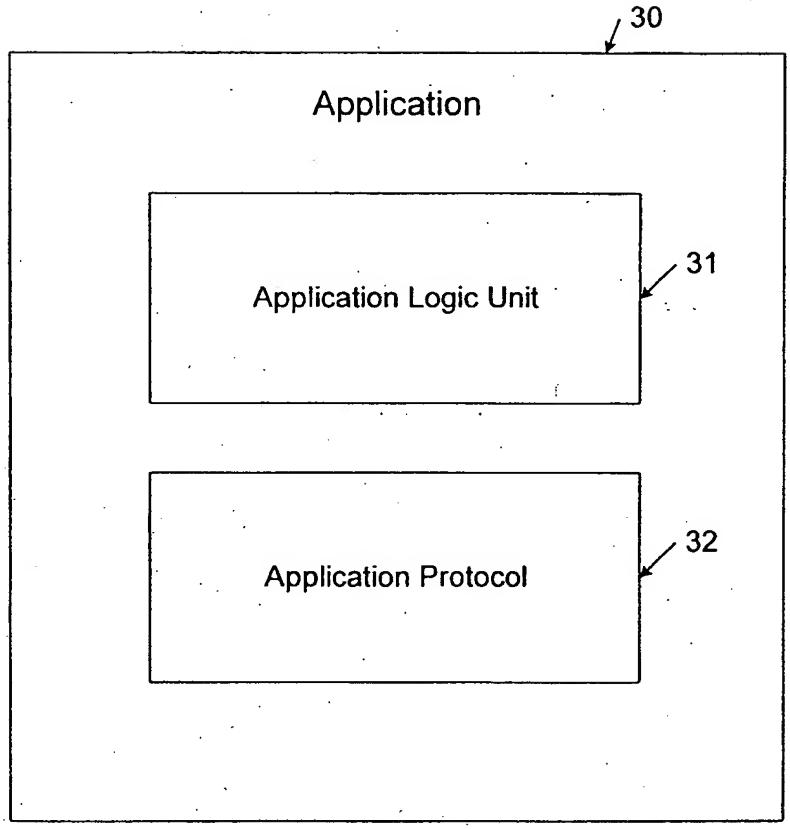

30
Application
Application Logic Unit
31
Application Protocol
32

Figure 3

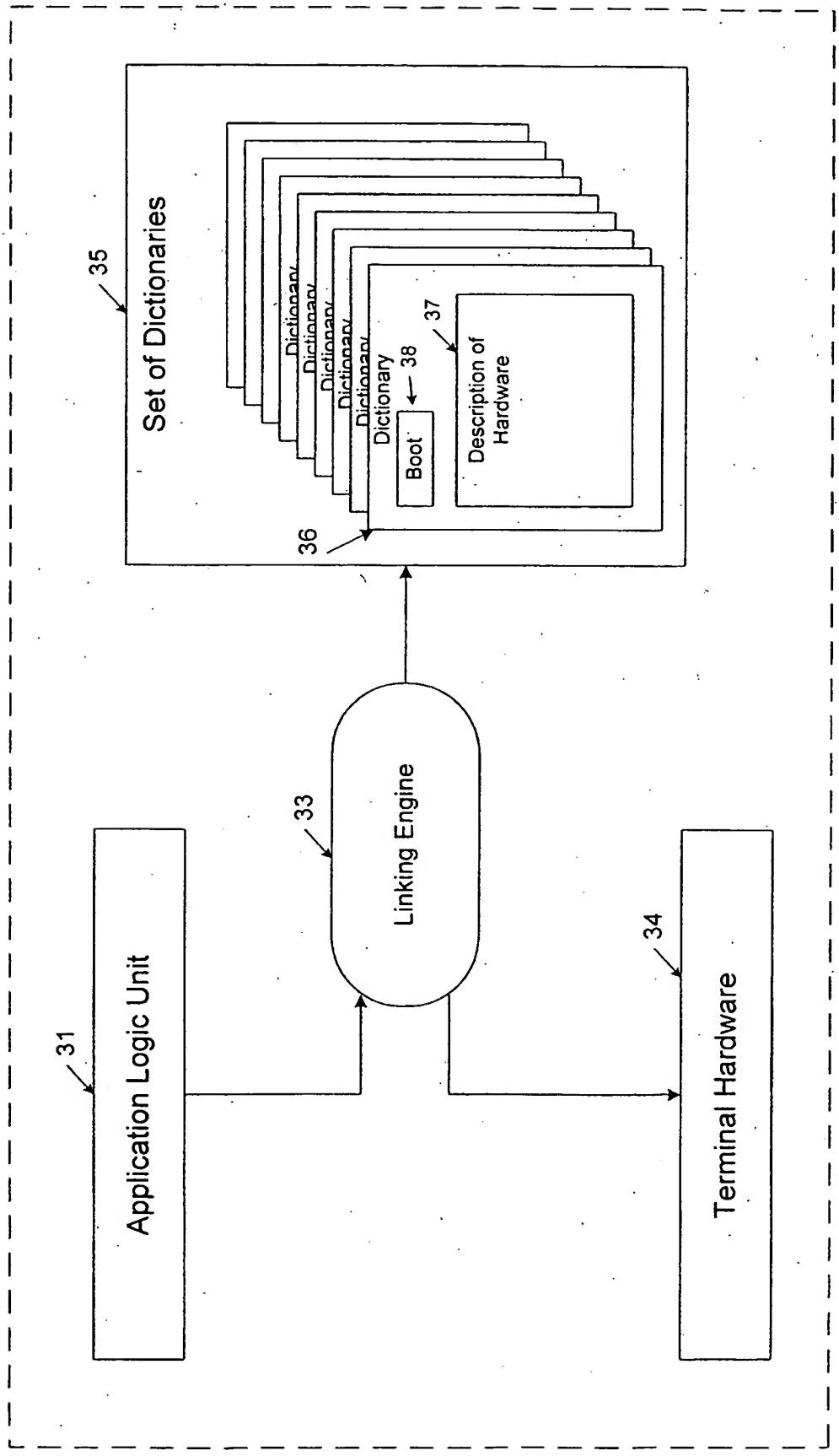

4/7
Application Logic Unit
31
Linking Engine
33
Terminal Hardware
34
Set of Dictionaries
35
Dictionary
Dictionary
Dictionary
Dictionary
Dictionary
Dictionary
36
38
Boot
37
Description of Hardware

Figure 4

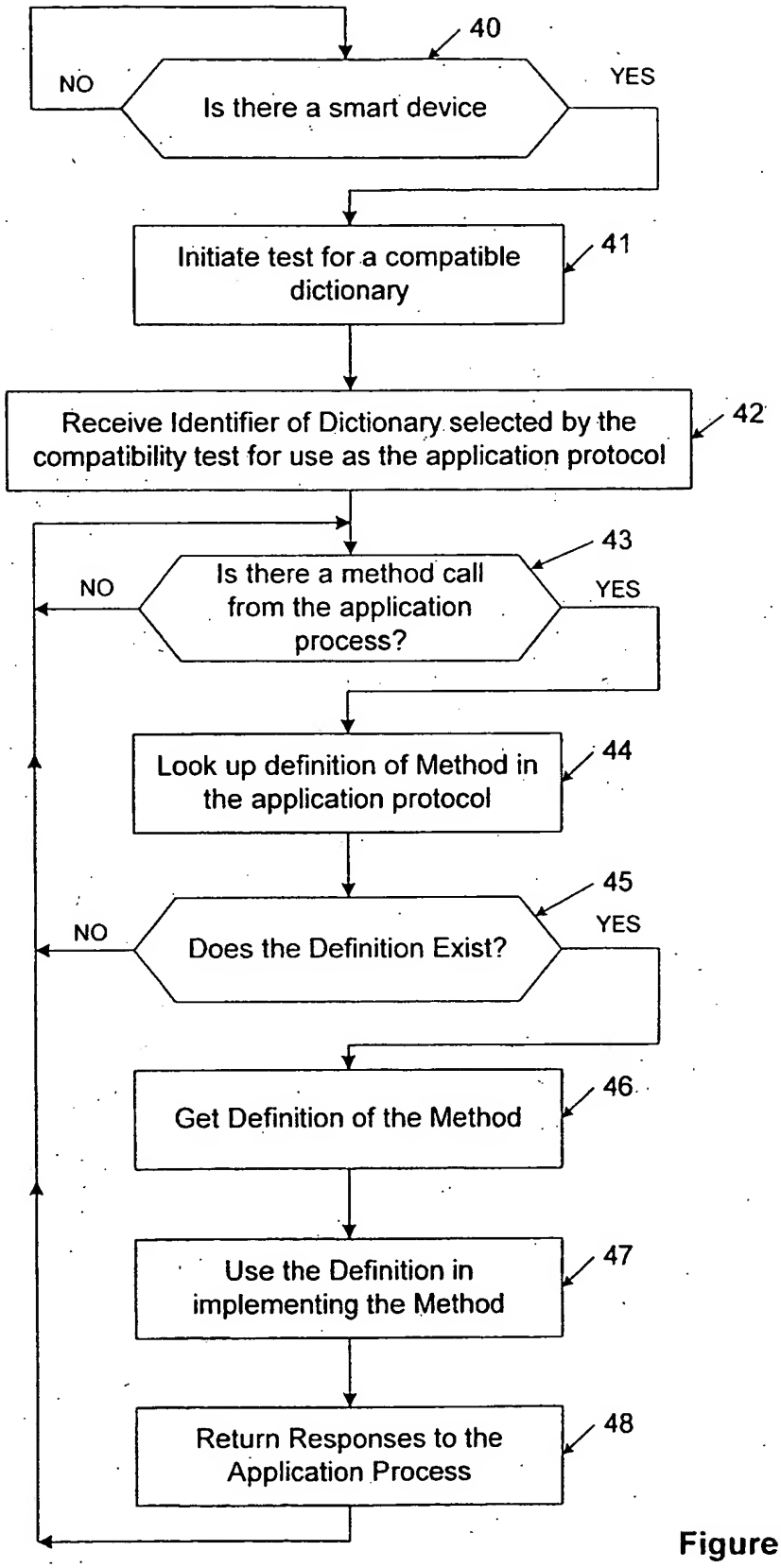

40
NO
Is there a smart device
YES
Initiate test for a compatible dictionary
41
Receive Identifier of Dictionary selected by the compatibility test for use as the application protocol
42
43
NO
Is there a method call from the application process?
YES
Look up definition of Method in the application protocol
44
45
NO
Does the Definition Exist?
YES
Get Definition of the Method
46
Use the Definition in implementing the Method
47
Return Responses to the Application Process
48

Figure 5

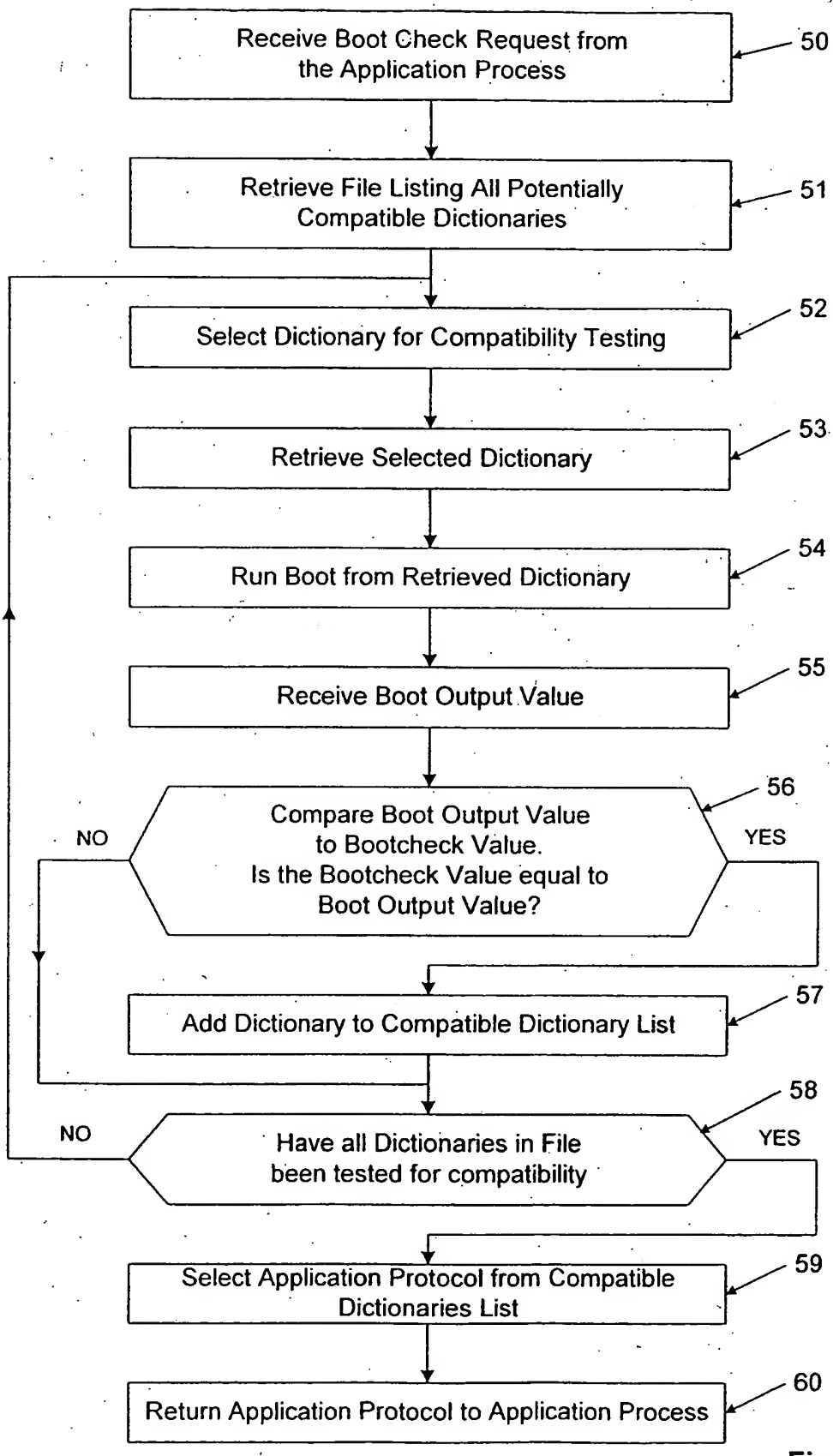

Receive Boot Check Request from the Application Process
50
Retrieve File Listing All Potentially Compatible Dictionaries
51
Select Dictionary for Compatibility Testing
52
Retrieve Selected Dictionary
53
Run Boot from Retrieved Dictionary
54
Receive Boot Output Value
55
Compare Boot Output Value to Bootcheck Value. Is the Bootcheck Value equal to Boot Output Value?
56
NO
YES
Add Dictionary to Compatible Dictionary List
57
Have all Dictionaries in File been tested for compatibility
58
NO
YES
Select Application Protocol from Compatible Dictionaries List
59
Return Application Protocol to Application Process
60

Figure 6

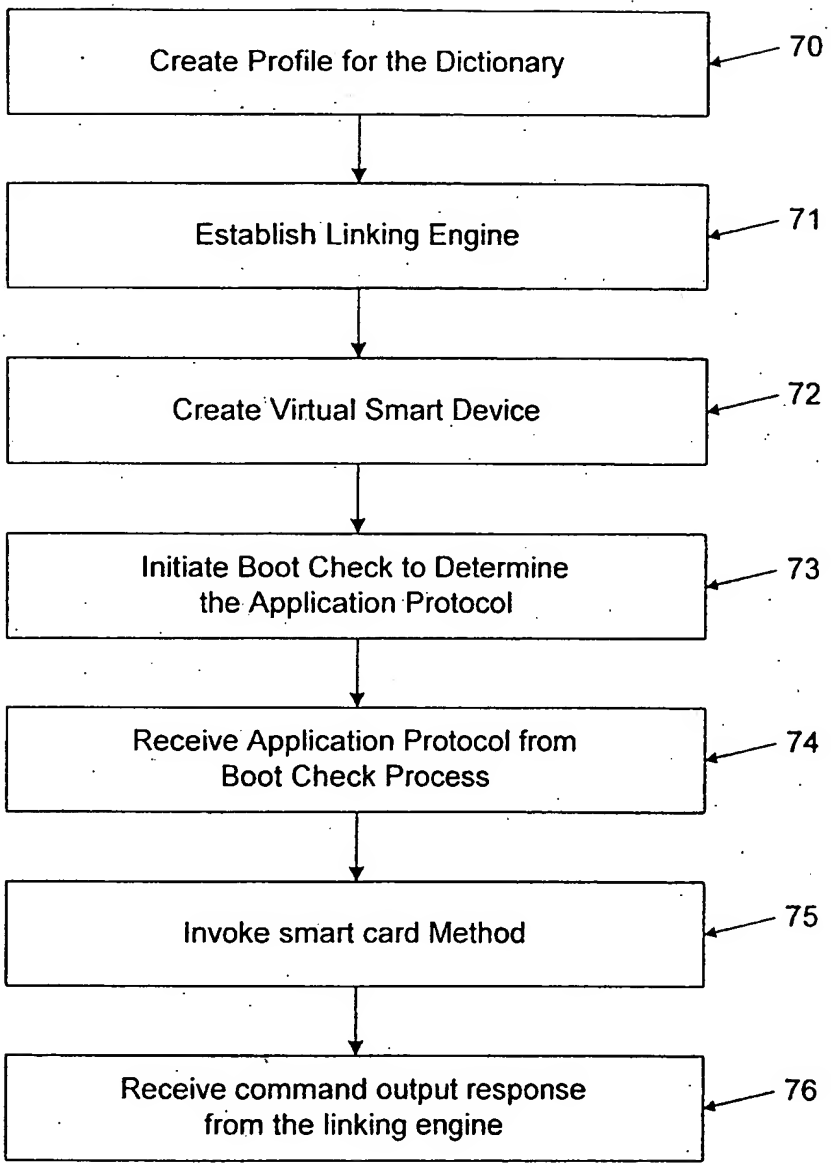

Create Profile for the Dictionary
70
Establish Linking Engine
71
Create Virtual Smart Device
72
Initiate Boot Check to Determine the Application Protocol
73
Receive Application Protocol from Boot Check Process
74
Invoke smart card Method
75
Receive command output response from the linking engine
76

Figure 7